Neural and statistical methods for the visualization of multidimensional data

by

Antoine Naud

A thesis submitted in conformity with the requirements
for the degree of Doctor in Technical Science
Katedra Metod Komputerowych
Uniwersytet Mikołaja Kopernika w Toruniu

Abstract

In many fields of engineering science we have to deal with multivariate numerical data. In order to choose the technique that is best suited to a given task, it is necessary to get an insight into the data and to “understand” them. Much information allowing the understanding of multivariate data, that is the description of its global structure, the presence and shape of clusters or outliers, can be gained through data visualization. Multivariate data visualization can be realized through a reduction of the data dimensionality, which is often performed by mathematical and statistical tools that are well known. Such tools are Principal Components Analysis or Multidimensional Scaling. Artificial neural networks have developed and found applications mainly in the last two decades, and they are now considered as a mature field of research. This thesis investigates the use of existing algorithms as applied to multivariate data visualization. First an overview of existing neural and statistical techniques applied to data visualization is presented. Then a comparison is made between two chosen algorithms from the point of view of multivariate data visualization. The chosen neural network algorithm is Kohonen’s Self-Organizing Maps, and the statistical technique is Multidimensional Scaling. The advantages and drawbacks from the theoretical and practical viewpoints of both approaches are put into light. The preservation of data topology involved by those two mapping techniques is discussed. The multidimensional scaling method was analyzed in details, the importance of each parameter was determined, and the technique was implemented in metric and non-metric versions. Improvements to the algorithm were proposed in order to increase the performance of the mapping process. A graphical user interface software was developed on the basis of those fast mapping procedures to allow interactive exploratory data analysis. Methods were designed to allows the visualization of classifiers decision borders.

Streszczenie

W wielu dziedzinach nauk inżynieryjnych mamy do czynienia z numerycznymi danymi wielowymiarowymi. Wybrać najbardziej odpowiedniej metody do rozwiązywania danego problemu często wymaga wglądu w dane aby je “zrozumieć”. Znaczna część informacji pozwalająca na zrozumienie danych wielowymiarowych, tak jak określenie ich globalnej struktury, obecności oraz kształtu klasterów lub danych odległych, może być uzyskana przez vizualizację tych danych. Wizualizacja danych wielowymiarowych może być zrealizowana za pośrednictwem redukcji wymiarowości danych, która bywa często wykonana przy pomocy dobrze znanych narzędzi matematycznych lub statystycznych. Przykładowymi takimi narzędziami są: Analiza Składników Głównych oraz Skalowanie Wielowymiarowe. Sztuczne sieci neuronowe znalazły wiele zastosowań w ostatnich latach, stanowią one dziś dojrzałą dziedzinę naukową. Poniższa praca analizuje zastosowanie istniejących algorytmów do wizualizacji danych wielowymiarowych. Przedstawiono przegląd szeregu istniejących neuronowych i statystycznych metod wykorzystanych do wizualizacji danych. Następnie porównano dwa wybrane algorytmy ze sobą z punktu widzenia wizualizacji. Wybrano sieć neuronową typu Samo-Organizującej się Mapy Kohonen’a i statystyczną metodę eksploracyjnej analizy danych Skalowanie Wielowymiarowe. Przedstawiono zalety i wady obu metod z punktu widzenia teoretycznego i praktycznego. Omówiono zachowanie topologii danych wynikające z tych rzutowań. Metoda skalowania wielowymiarowego została szczegółowo zanalizowana z podkreśleniem roli każdego jej elementu i parametru. Zaimplementowano wersje metryczne i niemetryczne tej metody. Zaproponowano różne rowiązania poprawiające skuteczność i szybkość działania procesu rzutowania. Program został wyposażony w graficzny interfejs dla użytkownika, dzięki czemu tworzy narzędzie do interakcyjnej eksploracji danych wielowymiarowych. Opracowano też metody wizualizacji granic decyzji klasyfikatorów.

Acknowledgements

I want to thank my supervisor, prof. Włodzisław Duch, for his advice, support and guidance.

The members of the Department of Computer Methods (KMK) gave me help and support, ranging from solving problems with computer and software, to providing me with their data and programs as well as running many calculations for me. I would like to thank in particular Rafał Adamczak, Karol Grudziński, Krzysztof Grąbczewski and dr Norbert Jankowski.

I would also like to thank prof. Noel Bonnet for his help and the collaboration we had during my stay in Reims and our meetings in Antwerp.

The most important “acknowledgement” goes to my wife Elżbieta.

Contents

List of Tables

List of Figures

List of abbreviations

AFN Auto-associative Feedforward Neural Network

ALSCAL Alternating Least squares SCALing

ANN Artificial Neural Network

ART Adaptive Resonance Theory

BMU Best Matching Unit

CA Correspondence Analysis

CCA Curvilinear Components Analysis

EDA Exploratory Data Analysis

GTM Generative Topographic Mapping

KDD Knowledge Discovery in Databases

KNN k-Nearest Neighbors

KYST Kruskal Young Shepard Torgerson Kruskal, Young and Seery. A merger of M-D-SCAL (5M) and TORSCA which combines best features of both, plus some improvements. Fortran IV code and manual available at StatLib. One of the first computer programs for multidimensional scaling and unfolding. The name KYST is formed from the initials of the authors.

LDA Linear Discriminant Analysis (or Fisher's Discriminant Analysis)

LSS Least Square Scaling

LVQ Learning Vector Quantization

MDS Multidimensional Scaling

MLP Multi-Layer Perceptron

MSA Multivariate Statistical Analysis

MST Minimal Spanning Tree

NLM Sammon's Non Linear Mapping

PCA Principal Components Analysis

PCO Principal Coordinates Analysis

PSN Principal Subspace Network

QR Algorithm computing the decomposition of any real matrix A into a product $Q \cdot R$, where Q is orthogonal and R is upper triangular, using Householder transformations.

RBF Radial Basis Function

SA Simulated Annealing

SMACOF Scaling by MAjorizing a COmplicated Function

SOM Self-Organizing Map

SVD Singular Values Decomposition

VQP Vector Quantization and Projection

Glossary of notation

$\{O_i\}$ a collection of objects studied and described by some measurements,

N the number of objects under consideration: $\{O_i, i = 1, \ldots, N\}$,

D the number of measurements performed on each object (or features),

$\mathcal{D}$ the D-dimensional data space (or feature space) in which the objects are described,

$\mathbf{X}$ a $[N \times D]$ matrix of the coordinates of the N objects O_i in the data space $\mathcal{D}$,

d the number of dimensions or features with which the objects are to be represented,

$\mathcal{M}$ the d-dimensional map (or representation) space in which objects are represented,

$\{P_i\}$ a set of points that represent the objects $\{O_i\}$ in the data space $\mathcal{D}$,

$\mathbf{x}_i$ a D-dimensional vector representing point P_i in the data space $\mathcal{D}$,

$\{p_i\}$ a set of points that represent the objects $\{O_i\}$ in the mapping space $\mathcal{M}$,

$\mathbf{y}_i$ a d-dimensional vector representing point p_i in the mapping space $\mathcal{M}$,

$\mathbf{Y}$ a $[N \times d]$ matrix of the coordinates of the N points P_i in the mapping space $\mathcal{M}$,

Y a $(N \times d)$ vector of the coordinates of the N points P_i, ordered point-wise,

N_t the total number of points taken into account in one MDS mapping,

N_m the number of points moving during the mapping ($N_m = N_t$ if no fixed point),

N_d the number of inter-point distances that are varying during the mapping process,

δ_{ij} dissimilarity of objects O_i and O_j, given as input or computed in the data space $\mathcal{D}$,

D_{ij} distance measure between points i and j in the input space,

d_{ij} distance measure between points P_i and P_j in the output space,

$\hat{d}_{ij}$ disparity that measures how well the distance d_{ij} “matches” the dissimilarity δ_{ij},

w_{ij} a weight associated to the pair of objects $\{O_i, O_j\}$,

$S(\mathbf{Y})$ Stress function value evaluated for the configuration held in matrix $\mathbf{Y}$,

$\nabla S(\mathbf{Y})$ gradient vector of Stress function S, evaluated at $\mathbf{Y}$,

$H_S(\mathbf{Y})$ Hessian matrix of Stress function S, evaluated at $\mathbf{Y}$,

α_S length of the move towards the opposite of the gradient of $S(\mathbf{Y})$, called step-size.

In Stress expressions, the notation $\sum\limits_{i<j}^{N}$ means $\sum\limits_{i=1}^{N-1}(\sum\limits_{j=i+1}^{N})$, and $\sum\limits_{i\neq j}^{N}$ means $\sum\limits_{i=1}^{N}(\sum\limits_{j=1}^{i-1}+\sum\limits_{j=i+1}^{N})$.

The author's software contribution

Programs for the Self-Organizing Maps:

- The SOM_PAK package [81] was used for the training of the SOM Neural Network, a few features were added, input of training parameters read from a separate text file,

- All the presented tools for map visualization were developed in the C language, except the U-matrix visualization tool that came with the SOM_PAK package.

Programs for Multidimensional Scaling:

- Metric and non-metric algorithms were entirely developed in the C++ language. Non-metric MDS was implemented firstly in the C language on the basis of Kruskal's KYST Fortran source [90]. Then it was translated into the C++ language including different proposed original improvements. Line command versions of those procedures were developed using the Borland C++ environment v.5.01,

- The graphical user interface `IMDS` allowing real-time visualization of mappings and interactive focusing on desired sub-sets was entirely developed using Borland C++ Builder v.4.0 development tool, it runs on Windows platform.

Chapter 1

Introduction

1.1 The need for data visualization

The rapid development of computers of the last decades allowed people to store and analyze an increasing number of data. Researchers more and more often have to deal with tens or hundreds of variable measurements that they obtained from the objects observed in their experiments. In some situations, the structure of the objects under consideration is well understood and a rather good model is known (e.g. a normal distribution). If no model of the data exist, some insight or understanding of the data can be gained by extracting from the data themselves information about their structure or patterns. A data-driven search for statistical insights and models is traditionally called *Exploratory Data Analysis* [134]. The nature of this information can be statistical (mean, variances, and so on) or more closely related to human observation capabilities (structures, clusters or dependencies). It is much easier for a human observer to detect or extract some information from a graphical representation of experimental data than from raw numbers. Visualization of multivariate data is hence often used to provide a synthetic view of patterns or clusters formed by the data, or to detect outliers [5]. This is why researchers, technicians or practitioners working with multidimensional data are very interested in data visualization software.

In order to introduce some notation, let us now consider the following general experimental situation: an observation is conducted on a finite number, say N, of objects $\{O_i, i = 1, ..., N\}$. The observer is taking a finite number, say D, of measurements of different nature on each object O_i. We assume here that all the measurements are taken successfully for all the objects (there is no missing value). The nature of the measurements taken, called here variables, are the same for all the objects, and each measurement gives a real number. The measurements can be arranged in a $[N \times D]$ real matrix $\mathbf{X}$ (each row of $\mathbf{X}$ correspond to an object and each column to a variable).

If only two variables are available (2-dimensional data, $D = 2$), a simple way to obtain a graphic representation of the objects is the scatter plot: on a plane spanned by 2 orthogonal axes $\overrightarrow{x}$ and $\overrightarrow{y}$ representing the 2 variables, we plot a point $P_i(x, y)$ with coordinates equal to the 2 measurements for object O_i, that is $x = x_1, y = x_2$. This simple idea can be extended to the case $D > 2$ by making scatter plots of all the possible pairs of variables (called *pair wise scatter plots*). But when the number of variables increases, the increasing number of scatter plots does not allow a synthetic view of the data by a human observer. Two alternative approaches can be distinguished to enable observation of high dimensional data on a graphic display: whether all the dimensions (or only the most important ones) are displayed together by some graphical means other than a scatter plot, or the number of dimensions is first reduced to 2 or 3 and the data in new dimensions are represented using a scatter plot.

At present, there exist a large number of different techniques allowing graphical representation of experimental data. The different visualization tools are designed depending on the different types of data available and on the goal of the visualization.

1.2 An overview of multivariate data visualization techniques

Let us now present briefly a number of methods that have been developed for the purpose of multivariate data visualization. The aim of this overview is not to provide an exhaustive panorama of the existing techniques, but to outline the variety of approaches.

- Feature selection and feature extraction methods: The dimensionality of the data can be reduced by choosing a few features that best describe our problem (feature selection), or by combining the features to create new ones that are more informative according to a given criterion (feature extraction). As mentioned in [121] and [120], the quality of the resulting mapping will depend on whether the chosen criterion for the new features is really satisfied by the data.

- Piecewise Linear Mapping: Data visualization through the minimization of piecewise-linear and convex criterion functions has been proposed in [17]. Algorithms similar to the linear programming methods minimize these functions based on a concept of clear and mixed dipoles. This general framework can be used to generate, among others visualizations based on Fishers discriminant analysis.

- Non Linear Mappings: The family of Multidimensional Scaling techniques [85], with e.g. Sammon's mapping [115] and its enhancements ([102] with a more general error criterion and the use of parameters that improve the algorithm's convergence) or variants facing the problem of large data sets by adjusting only a pair of points at each step (relaxation method) or by selecting a subset of points (frame method) [23] (see [129, p. 126]).

- Sequential non linear mappings: The triangulation method [92] performs a sequential mapping of high-dimensional points onto a plane. The idea is to map each point preserving exactly its distances to 2 previously mapped points, using the distances of the minimal spanning tree (MST) of the data. This method leads to an exact preservation of all the distances of the MST, but it is sensitive to the order in which the points are mapped. The equal-angle spanning tree mapping [138] is similar to the triangulation method, with the difference that it leads to the preservation of the minimal spanning tree itself (that is, input and mapped data have the same MST).

- Projection pursuit: Projection pursuit [50] is a technique that seeks out "interesting" linear projections of multivariate data onto lines or planes. The best projection line or plane is the one for which an "interestingness" or projection index is maximized (by a classical optimization technique). Friedman and Tukey proposed an index of interestingness purposely designed to reveal clustering. This index was defined as a product of a measure of the spread of the data by a measure of data local density after projection. This leads to projections tending to concentrate the points into clusters while, at the same time, separating the clusters. This is similar to Fisher's discriminant's heuristic, but without making use of the data class information. This technique suffers from the limitations of any linear mapping, having difficulty in detecting clustering on highly curved surfaces in the data space.

- Grand tour method: A human can observe simultaneously at most three dimensions, so data visualization in a 3-dimensional space is useful and provide more information than in 2 dimensions. Grand tour methods [19], a part of the computer graphical system Xgobi [124], allow rotating graphs of three variables. This method is based on the simple idea of moving projection planes in high dimensional data spaces. Projecting high dimensional data onto these planes in rapid succession generates movies of data plots that convey a tremendous wealth of information.

- The biplot: The biplot devised by Gabriel [54] [61] is closely related to the scatter plot of the first principal components, but additionally to the N points plotted for the N objects or observations, it contains D points representing the D dimensions or variables used. Some of the techniques of Correspondence Analysis produce similar kinds of plots. The term biplots is also used in [61] to name a family of techniques (including MDS, CA or PCA) leading to a graphical representation which superimpose both the samples and the variables on which the samples are measured. The 'bi' in biplots arises from the fact that both the samples and the variables are represented on the same graph. In the family of multidimensional scaling techniques, *unfolding* is one with a purpose of producing such plots containing "subject" points and "stimulus" points.

- Cluster analysis techniques: This last category of methods differs from the majority of other methods by the fact the class information of the data points is mainly used. The basic objective in cluster analysis is to discover natural groupings of the objects. Searching the data for a structure of "natural" groupings is an important exploratory technique. Groupings can provide an informal means for assessing dimensionality, identifying outliers, and suggesting interesting hypotheses concerning relationships. The techniques described here are always accompanied by a graphical representation of the groupings. Grouping is done on the basis of similarities or distances (dissimilarities), so the input required are similarity measurements or data from which similarities can be computed. It is clear that meaningful partitions depend on the definition of *similar* as well as on the grouping technique.

- Special pictorial representations: Some techniques have been designed to display multivariate data in 2-dimensional graphics directly (that is without dimensionality reduction). We can mention here multiple 2-dimensional scatter plots (for all the pairs of variables), Andrews plots [2] or Chernoff faces [24], which are discussed in [46]. Categorical multivariate data can also be represented on a synthetic scatter plot as proposed in [72, pp. 147-150].

There exist very few comparisons of different projection algorithms in the literature. Such attempts have been presented in [15], [120] and [121]. In this last paper, Siedlecki et al. presented an attempt to systemize mapping techniques, which can be summarized as follows:

- Linear vs. Non linear transformations: A linear transformation is a transformation for which there is a linear relationship between the input and output data of this transformation, that is, the mapping is executed by a matrix multiplication. Within this category, we can distinguish principal components based methods, Fisher's discriminant's based methods, least squares and projection pursuit methods.

- Analytic vs. Non analytic transformations: Analytic transformations map every point in the D-dimensional data space, whereas non analytic transformations do not provide any analytical expression that would tie the coordinates of a D-dimensional data point with the coordinates of its planar representative.

The authors noted that the separation into linear and non-linear transformations corresponds almost exactly to the analytic and non-analytic transformations. We will see in the following chapters of this work that this correspondence does not hold for the methods available today, especially when considering neural networks. A last category can be added to the previous ones, which is **supervised** vs. **unsupervised** methods, that is mappings that make use or not of the data class information.

1.3 The paradigm of data visualization by dimensionality reduction

We are concerned in this part with the techniques allowing the visualization of high dimensional data through a reduction of its dimensionality. In order to obtain a satisfying graphical representation of the objects under consideration, the dimensionality reduction of the objects must preserve the information *that is important to the observer for its analysis*. The important information is often a distance or similarity measure, or else inter-point variance. The search for this mapping or projection is called here the *dimensionality reduction problem (DR)*, which we formulate as follows:

Let $\{x_{ij}, j = 1,...,D\}$ be a series of D experimental measurements taken on object O_i. The measurements performed on a set of N objects are arranged in a $[N \times D]$ matrix X called *data matrix*. Each object O_i can be seen as a point P_i in a D-dimensional metric space, and is described by a D-dimensional vector $\mathbf{x}_i = (x_{i1} \cdots x_{iD})^T$. The DR problem consists in looking for a configuration of N points $\{Q_i, i = 1,...,N\}$ in a space of dimensionality $d < D$ in which each point Q_i will represent an object O_i in the target d-dimensional space, so as to satisfy an *information criterion IC*. Let $\mathbf{Y}$ be the matrix of the coordinates of the N points $\{Q_i\}$ constructed in the same way as matrix $\mathbf{X}$.

$$\left(\mathbf{X} = [x_{ij}] = \begin{bmatrix} x_{11} & \cdots & x_{1D} \\ \vdots & \ddots & \vdots \\ x_{N1} & \cdots & x_{ND} \end{bmatrix}, IC \right) \quad \Longrightarrow \quad \mathbf{Y} = [y_{ij}] = \begin{bmatrix} y_{11} & \cdots & y_{1d} \\ \vdots & \ddots & \vdots \\ y_{N1} & \cdots & y_{Nd} \end{bmatrix} \tag{1.1}$$

The fact that the dimensionality of the points is reduced involves an unavoidable loss of information, and the criterion IC is seldom fully satisfied. The method used has to be such that the information contained in the data *that is important to the user* is preserved as much as possible. Various methods employed to compute the reduced dimensions will differ on what kind of information will be retained. (The information criterion IC can be for example: inter-point Euclidean distances preservation, the rank orders of inter-point distances preservation, the presence of clusters of data or variances of variables).

The need for dimensionality reduction also arises from more practical reasons. Although present computers have still growing memory and computing capabilities, software needs are always increasing, so that dimensionality reduction is helpful in the following computer tasks:

- DR allows to reduce the amount of memory needed to store information represented by vectors, such as images or videos,
- DR makes easier and faster further manipulation of the data,
- DR saves computation time spent to process or analyze the data (in subsequent classification or clustering tasks)
- DR can improve the analysis performances by reducing the effect of noisy dimensions.

In pattern recognition, where the data have to be used as input to processing procedures whose computation time can grow importantly with the number of dimensions, DR is necessary to enable the use of certain techniques. This is especially crucial for data such as images or acoustic signals because the number of such objects under analysis is usually high. The problem called *curse of dimensionality*, appearing when the ratio of the number of points to the number of dimensions is too low can also be avoided by DR [43].

1.4 Aims of the research

The main objective of the thesis is to compare and apply SOM and MDS algorithms as tools for multivariate data visualization. When desirable for this purpose, some improvements of the existing algorithms were proposed. This objective is divided into the following points:

- Study SOM algorithm and analyze the resulting mappings,
- Study MDS algorithm and analyze the resulting mappings,
- Compare SOM and MDS mappings from the point of view of data topology preservation,
- Improve and apply MDS mapping to the interactive visualization of multivariate data and classifiers decision borders.

1.5 Structure of the thesis

Before focusing on the two main algorithms under the scope of this work (Self-Organizing Maps and Multidimensional Scaling), the well known method of Principal Components Analysis is presented in the next chapter (chapter 2). A first reason for this is that this method performs a linear dimensionality reduction, whereas the other two are non-linear, so it is interesting to study its specificity. A second reason is that PCA is often used to initialize the other two methods for which the better the initialization, the better the final result. Then the two non-linear mapping methods are presented as iterative methods that start from an initial guess and improve it step by step to lead to the result. The Self-Organizing Maps algorithm is presented in chapter 3 among other artificial neural networks used in data visualization. The main features and limitations of this approach to data visualization are presented from a practical point of view. A few data sets are visualized to illustrate in which manners the algorithm allows the display of data. Chapter 4 is devoted to the Multidimensional Scaling techniques, where details are given on two of the most popular existing implementations of Least Squares Scaling: Sammon's non-linear mapping and Kruskal's non-metric MDS. Practical limitations are described, and various improvements are proposed to fasten the calculations and optimize the results. The following chapter 5 contains a short comparison of the SOM and MDS methods from a practical point of view. Then MDS is applied to various real-life data visualization tasks in chapter 6. Several tools helping the interactive exploration of databases are used to visualize some medical data sets, and finally MDS is applied to the visualization of classifiers decision boundaries. A conclusion chapter (7) summarizes the most important results of this work and lists promising possible further developments.

Chapter 2

Linear mapping by Principal Components Analysis

The use of principal components analysis to solve the dimensionality reduction problem can be summarized as follows: first, a linear transformation of the data points is searched so that the variances of the points in the transformed dimensions are in decreasing order of magnitude. Then a selection of the first d dimensions allows to obtain a set of points with reduced dimensionality that preserve the variance optimally. Hence the criterion used is the preservation of the variance on the variables, making the underlying assumption that important information lies in the variances. It can be shown that if a set of input data has eigenvalues $(\lambda_1, \lambda_2, ..., \lambda_n)$ and if we represent the data D coordinates on a basis spanned by the first d eigenvectors, the loss of information due to the compression is $E = \sum_{i=d+1}^{D} \lambda_i$.

The methods described in this section are called *linear methods* because the variables searched are linear combinations of the original variables. The statistical techniques by which such a reduction of data is achieved are known collectively as *factor analysis*[1]. In contrast to *iterative methods* that will be the object of the following Chapters 3 and 4, linear methods are called *direct methods* because a solution to the dimensionality reduction problem is derived analytically. Whereas in direct methods a solution is computed in one step, in iterative methods a starting guess is first computed and a number of approaching steps are taken to find a solution. It must be noted that the division made here between direct and iterative methods holds for the precise algorithms that will be discussed in Chapters 3 and 4 of this thesis, but not for all the methods bearing the names PCA or MDS. For example in Chapter 4 devoted to multidimensional scaling methods, a direct method is referred to (Classical scaling or Gower's PCO [60]). Inversely, Oja's neural network version of PCA [105] is, as any Artificial Neural Network, an iterative algorithm.

Linear methods have a long history (Principal Components Analysis was introduced in [107] [70]), they are well known and have been successfully applied in various fields of science. They are Multivariate Statistical Analysis methods using tools from matrix algebra, hence their presentation in the formalism of matrix algebra.

The principal components of a set of points are linear combinations of their variables that have special properties in terms of variances. The first Principal Component (PC) is the normalized linear combination of the variables with maximum variance, the second PC is the normalized linear combination of the variables perpendicular to the first PC with maximum variance, and

[1] The object of factor analysis is to find a lower-dimensional representation that accounts for the correlations among the features, whereas the object of principal components analysis is to find a lower-dimensional representation that accounts for the variance of the features. [43, p. 246]

so on. So if we want to obtain a dimensionally reduced set of points that preserve as much as possible the variances of the original set of points, an optimal choice is to take for $\mathbf{Y}$ the d first PCs of $\mathbf{X}$. The principal components of a set of points can be computed in several ways, but two main approaches are generally distinguished:

- The first approach has the longest history in multivariate statistical analysis and is the best known. It consist in computing first the between variables covariance or correlation matrix $\mathbf{S}$ of matrix $\mathbf{X}$, and then taking the spectral (or eigenvalue) decomposition of $\mathbf{S}$. Dimensionality reduction is obtained by projections of the points $\{P_i\}$ (the rows of $\mathbf{X}$) on the d first principal components of $\mathbf{S}$.

- In the second approach, the principal components of the points are obtained from the Singular Value Decomposition (SVD) of matrix $\mathbf{X}$. There is no need here to compute a covariance or correlation matrix. The main advantage of the SVD approach over the spectral decomposition approach lies in its better numerical stability [112, p. 290]. The price for this is a greater need for memory space and computation time (in the SVD process a $[N \times N]$ matrix has to be stored). This last argument can make the SVD approach unpractical for dimensionality reduction problems where the number N of objects is very large.

In the following chapters, initializations of codebook or configuration were computed using the SVD approach for the reasons advocated here above. The computational details of both approaches are given in the following two sections.

2.1 Spectral decomposition of the correlation matrix

The spectral decomposition (or eigendecomposition) of the correlation matrix is performed through a tridiagonalization of the correlation matrix followed by a QL algorithm. The complete process of reduction of the dimensionality of a set of N data points (D-dimensional) forming a $[N \times D]$ matrix $\mathbf{X}$ to a set of N points (d-dimensional) through spectral decomposition of the correlation matrix consists of the following steps:

1. Compute the correlation matrix $\mathbf{S}$:
   (a) Center the data points at the origin (ie. remove column means out of $\mathbf{X}$):
   $\mathbf{X} = \mathbf{X} - \frac{1}{N}(\mathbf{1} \cdot \mathbf{1}^T) \cdot \mathbf{X}$, where $\mathbf{1}$ is a N-dimensional vector of ones, [2]
   (b) Normalize the column standard deviations to get standardized matrix $\mathbf{X}_{STD}$:
   $\mathbf{X}_{STD} = \mathbf{X} \cdot \mathbf{D}^{-1/2}$, where $\mathbf{D} = diag(\mathbf{X}^T \cdot \mathbf{X})$ is the diagonal matrix of variances,
   (c) The correlation matrix $\mathbf{S}$ is the inner product of matrix $\mathbf{X}_{\mathbf{S}TD}$ by itself:
   $\mathbf{S} = \mathbf{X}^T_{\mathbf{S}TD} \cdot \mathbf{X}_{\mathbf{S}TD}$, ($\mathbf{S}$ is a $[N \times N]$ matrix).

2. Compute the spectral decomposition of matrix $\mathbf{S}$ in the following two steps:
   (a) Transform the symmetric matrix $\mathbf{S}$ to a tridiagonal matrix $\mathbf{S}_T$ through a reduction process consisting of $N-2$ Householder orthogonal transformations of $\mathbf{S}$:
   $\mathbf{S}_T = \mathbf{P}^{(N-2)} \cdot \mathbf{P}^{(N-1)} \cdot \ldots \cdot \mathbf{P}^{(1)} \cdot \mathbf{S} \cdot \mathbf{P}^{(1)} \cdot \ldots \cdot \mathbf{P}^{(N)}$, where $\mathbf{P}^{(k)}$ is orthogonal.
   (b) Extract the eigenvalues and eigenvectors of the tridiagonal matrix $\mathbf{S}_T$ by the tridiagonal QL algorithm with implicit shifts:
   $\mathbf{S}_T = \mathbf{C} \cdot \mathbf{D} \cdot \mathbf{C}^T$, where $\mathbf{D}$ is a diagonal matrix containing the eigenvalues and $\mathbf{C}$ contains the eigenvectors.

[2] The raised dot $\cdot$ denotes the matrix product and the superscript ^{T} denotes the transpose of a matrix.

3. Sort the eigenvalues in decreasing order and reorder eigenvectors correspondingly.

4. The projections of the data points $\{P_i\}$ (the rows of $\mathbf{X}$) on the first d eigenvectors of $\mathbf{S}_T$ ($\mathbf{C}_d$ is made of the first d columns of $\mathbf{C}$) give the matrix $\mathbf{Y}$ of the N points, d-dimensional: $\mathbf{Y} = \mathbf{X} \cdot \mathbf{C}_d$.

2.2 Singular Value Decomposition of the data matrix

The Singular Value Decomposition (SVD) of a data matrix is performed here through its reduction to a bidiagonal form by Householder transformations followed by a QR algorithm to find the eigenvalues [56]. The complete process of dimensionality reduction of a set of N data points (D-dimensional) forming a $[N \times D]$ matrix $\mathbf{X}$ to a set of N points (d-dimensional) through Singular Value Decomposition of the data matrix consists of the following steps:

1. Center the data points at the origin (i.e. remove the column means out of $\mathbf{X}$): $\mathbf{X} = \mathbf{X} - \frac{1}{N}(\mathbf{1} \cdot \mathbf{1}^T) \cdot \mathbf{X}$, where $\mathbf{1}$ is a N-dimensional vector of ones.

2. Compute the Singular Value Decomposition of matrix $\mathbf{X}$ in the following two steps:

   (a) Matrix $\mathbf{X}$ is reduced to its upper bidiagonal form $\mathbf{X}_B$ (i.e. $\mathbf{X}_B[i,j] \neq 0$ only for $j = i$ or $j = i+1$) by Householder reflections from the left and the right: $\mathbf{X}_B = \mathbf{P}^{(N)} \cdot \ldots \cdot \mathbf{P}^{(1)} \cdot \mathbf{X} \cdot \mathbf{Q}^{(1)} \cdot \ldots \cdot \mathbf{Q}^{(N-2)}$, where $\mathbf{P}^{(k)}$ and $\mathbf{Q}^{(k)}$ are unitary matrices: $\mathbf{P}^{(k)} = \mathbf{I} - 2\mathbf{x}^{(k)}\mathbf{x}^{(k)^T}, k = 1, \ldots, D$ and $\mathbf{Q}^{(k)} = \mathbf{I} - 2\mathbf{y}^{(k)}\mathbf{y}^{(k)^T}, k = 1, \ldots, D - 2$.

   (b) A variant of the QR algorithm is used to diagonalize $\mathbf{X}_B$, computes the singular value decomposition of the bidiagonal form and transforms it back to obtain: $\mathbf{X}_B = \mathbf{N} \cdot \mathbf{D}_\alpha \cdot \mathbf{M}^T$, where $\mathbf{N}, \mathbf{M}$ are orthogonal matrices and $\mathbf{D}_\alpha$ is diagonal.

3. The rank d approximation of $\mathbf{X}$ (the first d left singular vectors multiplied by the first d singular values) gives the matrix $\mathbf{Y}$ of the N d-dimensional points coordinates: $\mathbf{Y} = \mathbf{N}_{(d)} \cdot \mathbf{D}_{\alpha(d)}$,

2.3 Experimental comparison of the two approaches

Experimental mappings performed by spectral decomposition of the covariance matrix and Singular Value Decomposition of the data matrix were conducted in order to evaluate the practical importance of the choice of the method. We employed the ready-to-use procedures from ”Numerical Recipes in C” [110, §2.6, §11.2 and §11.3]. A first observation of our experiments applying those two procedures on several data sets is that for the SVD approach, it is better to use double precision in floating number machine representation whereas this makes almost no difference in the case of spectral decomposition approach. The two algorithms were compared from the viewpoints of first their numerical accuracy (that is how much of variance is collected on the first principal axes) and second the displays of the resulting two-dimensional configurations of points).

2.3.1 Variance on the principal axes

As it was mentioned above, Principal Components Analysis allows extracting by linear combinations new features with maximum variances. A comparison of variances along principal axes obtained by the two methods presented above will therefore be a good indicator of efficiency for each method, the best one aggregating more variance in the first principal axes. Such an experiment was performed on a real life data set that has variance quite uniformly distributed among the features. The data used is from the Wisconsin Breast Cancer Database that was obtained from the University of Wisconsin Hospitals, Madison from Dr. William H. Wolberg [94], and is available at the UCI repository [16]. As this data set may be used in further experiments in this work, this experiment is a first statistical analysis that gives useful insight into the data. The data set is made of 699 cases (but only 463 single data), each belonging to one of two classes (`benign` or `malignant`). Each case is a patient described by 9 numerical attributes ranging from 1 to 10 (clump thickness, uniformity of cell size, uniformity of cell shape, marginal adhesion, single epithelial cell size, bare nuclei, bland chromatin, normal nucleoli, mitoses). The 16 missing attribute values were estimated by attribute averaging per class (see Chapter 6). Table 2.1 presents the distributions of variances among features for the original data set (left part), for the data set obtained from PCA by Spectral Decomposition (center part) and for the data set obtained from PCA by Singular Value Decomposition (right part). It can be seen that the projections of the data on each of the 3 first Principal Components computed by the SVD approach have larger variances (48.54; 5.10; 4.27) that the ones obtained from the SpD (47.00; 4.21; 4.15). This shows that SVD better captures components with large variance (combinations of features are more optimal), hence should it be preferred for reasons of accuracy.

`breast` data set			PCA by SpD			PCA by SVD		
Feature index	Var.	Percent	Eigen value	Var.	Percent	Singular value	Var.	Percent
6	13.10	18.62 %	5.89	47.00	66.80 %	184.20	48.54	68.99 %
8	9.31	13.23 %	0.78	4.21	5.99 %	59.69	5.10	7.24 %
2	9.30	13.22 %	0.54	4.15	5.90 %	54.61	4.27	6.06 %
3	8.82	12.54 %	0.46	3.53	5.02 %	23.67	3.13	4.44 %
4	8.14	11.57 %	0.38	2.86	4.06 %	46.74	2.74	3.89 %
1	7.92	11.25 %	0.31	2.75	3.91 %	43.77	2.42	3.43 %
7	5.94	8.44 %	0.29	2.74	3.90 %	41.10	1.77	2.52 %
5	4.90	6.96 %	0.26	2.27	3.23 %	33.36	1.59	2.26 %
9	2.94	4.17 %	0.09	0.84	1.19 %	35.21	0.80	1.14 %
Total	70.36	100.00 %		70.36	100.00 %		70.36	100.00 %

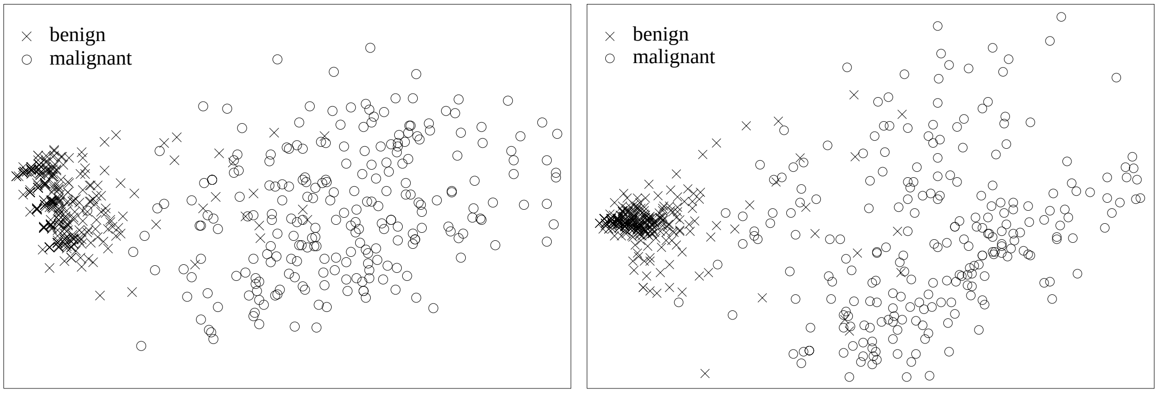
Table 2.1: Distributions of variance among features for the `breast` data set.

2.3.2 Visual comparison of configurations

The two configurations of the `breast` data set obtained previously from PCA using SpD and SVD are shown in figure 2.1. We see that the configurations are different enough, even after proper symmetry and rotation using, e.g. a Procrustes analysis (see §4.2.5). We conclude that even small differences of variance distribution among the features lead to configurations noticeably different. For this reason, we will use the SVD approach to compute PCA mappings through the remaining of this work.

2.3.3 Limitations of PCA dimensionality reduction

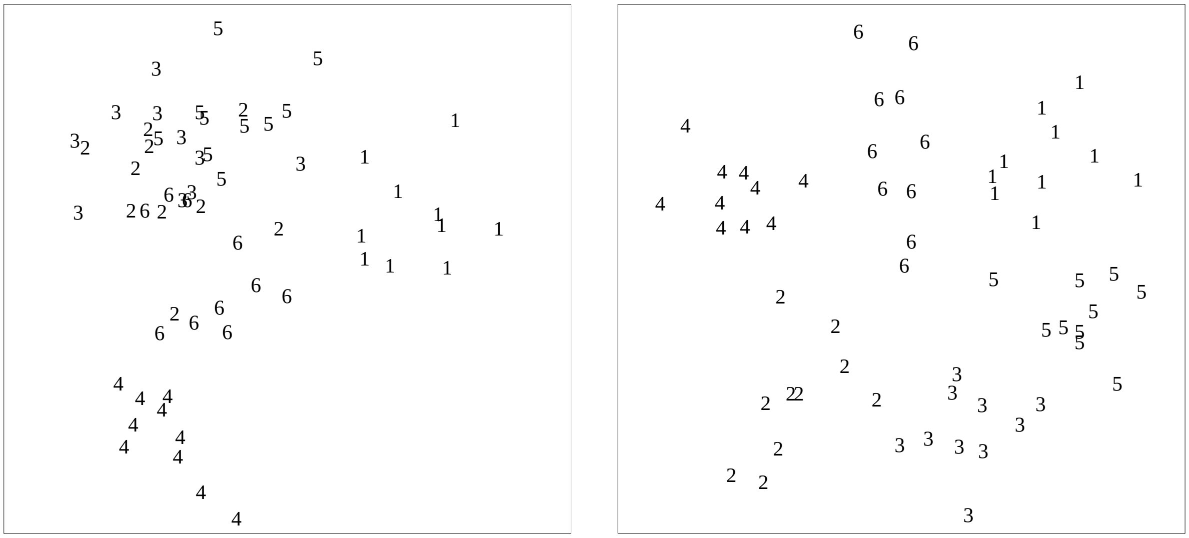
The main limitation of dimensionality reduction by PCA is that it performs a linear mapping. This means that this method is not suited to the visualization of data sets that are structured in a



(a) Spectral Decomposition of correlation matrix. (b) Singular Value Decomposition of data matrix.

Figure 2.1: Visualization of `breast` data set by Principal Components Analysis.

non-linear way (that is data sets for which only a non-linear transformation will provide a display that reflects the data structure). A good illustration of this property is to map using PCA an artificial data set structured specially in a non-linear way, for example the data set called `simplex5`, constructed as follows: First generate the vertices of a simplex in a 5-dimensional space, so that the inter-vertex distances are all equal to 1. Second generate 10 points in 6 Gaussian distributions, each centered at one vertex, with null covariances and identical variances in all dimensions equal to 0.3, in order to avoid overlap between the 6 clouds of points, labeled by numbers from 1 to 6. This data set is intrinsically 5-dimensional and the symmetry of the 6 groupings positions cannot be rendered on a 2-dimensional display using a linear mapping method such as PCA, whereas this is achieved by a non linear mapping method as MDS. The two displays are shown in figure 2.2. Another known problem of PCA dimensionality reduction is its sensitivity to the presence of outliers [112]. PCA is based on variances, and an outlier is an isolated point that artificially increases the variance along a vector pointing towards it. Taking an eigendecomposition of a robustly estimated covariance matrix can reduce this effect.



(a) Linear mapping using Principal Components Analysis: 3 groupings ($n°2,3$ and 5) are mixed. (b) Non linear mapping using Multidimensional Scaling: the 6 groupings are well separated.

Figure 2.2: Linear and non linear mappings for the visualization of `simplex5` data set.

2.4 Neural network implementations of PCA

See section 3.1 on neural networks for a general presentation of this family of techniques. Some neural networks have been explicitly designed to calculate Principal Components[3]. First a single neuron was implemented by Oja [103] that used a modified Hebbian learning rule (called "Oja's rule"). The Hebbian learning rule expresses the idea that the connection strength (or weight) between two neurons should be increased if the neurons are activated together, taking a weight increase proportional to the product of the simultaneous neurons activations. The Hebbian rule for one neuron is:

$$\Delta w_i = \alpha x_i y \tag{2.1}$$

where α is the learning rate, x_i is the i-th input to the single output neuron and y the output of this neuron. The output y sums the input in the usual fashion:

$$y = \sum_{i=1}^{d} w_i x_i \tag{2.2}$$

Oja proposed the following modified Hebbian learning rule with weight decay [4] :

$$\Delta w_i = \alpha (x_i y - y^2 w_i) \tag{2.3}$$

Using this rule, the weight vector $\mathbf{w}$ will converge to the first eigenvector. Then Oja [104] proposed a neural network based on this principle, in order to perform a Principal Components Analysis. By adding $d-1$ other neurons interacting between themselves we will find the other PCs. The method called Oja's subspace algorithm is based on the rule:

$$\Delta w_{ij} = \alpha (x_i y_j - y_i \sum_{k=1}^{d} w_{kj} y_k) \tag{2.4}$$

The weights have been shown to converge to a basis of the Principal Subspace. This neural network called the Principal Subspace Network (PSN) performs directly the mapping from input data space to the subspace spanned by the d first Principal Components[5], but without indication on the principal components order. Other ANN implementations of PCA that does not suffer from this problem have been proposed: Oja's Weighted Subspace [105] and Sanger's Generalized Hebbian Algorithm [116].

[3] Two main motivations for ANN-based PCA are i) a neural network can easily learn a very large data set, when the SVD approach can be unpractical for memory and time requirement reasons, and ii) a neural network can learn on-line new data as they arrive.

[4] A major difficulty with the simple Hebb learning rule is that unless there is some limit on the growth of the weights, the weights tend to grow without bound. Hence the role of the second part (the subtracted weight decay) is to re-normalize the weight vector at each iteration.

[5] This is the most interesting feature of this neural network, because available Singular Value Decomposition routines can handle quite large data matrices $\mathbf{X}$.

Chapter 3

The neural networks approach: Self-Organizing Maps

3.1 What are Artificial Neural Networks?

Artificial Neural Networks (ANN) are algorithms inspired by biology. The idea is to build systems that reproduce the structure and functioning of the brain neurons. Research in this field began in the 1940s, with the works of McCullogh and Pitts [97], followed by Hebb [66], Rosenblatt [114] and Widrow and Hoff [141]. An Artificial Neural Network can be described as a set of interconnected adaptive units generally organized in a layered structure [125]. A description of such a structure is presented in figure 3.1.

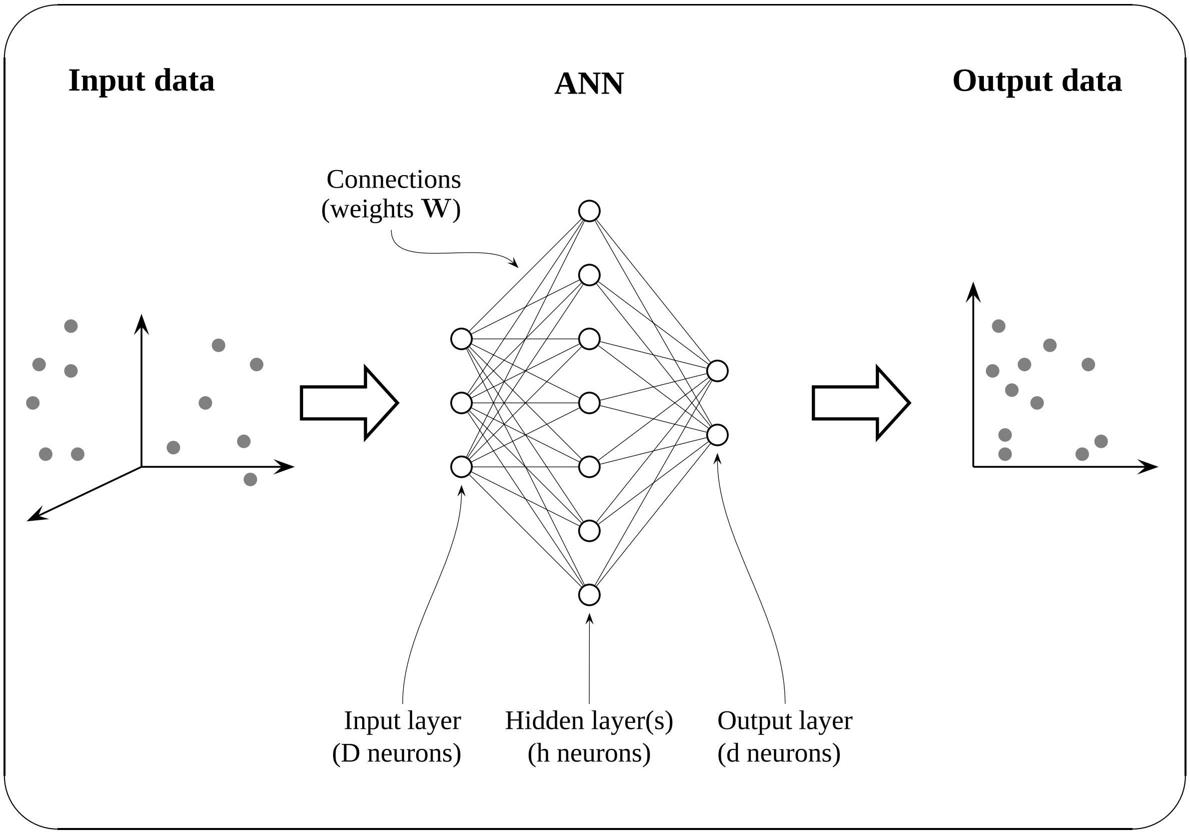


Figure 3.1: Generic structure of an Artificial Neural Network

The adaptation process (or learning) consist of a repeated presentation of some data to the network so long as it adapts its own inner parameters (or weights **W**) to acquire a representation of the data. The weight distribution over the network is the obtained data representation. From a technical point of view, ANN consist of a great number of simple computing elements which are connected to each other via unidirectional connections. A common case is to take a set of perceptrons [114] as units and arrange them in layers, forming the multilayer perceptron (MLP). From a statistical point of view, ANN are non-parametric models and make rather weak assumption of the underlying structure of the data.

3.2 Artificial Neural Networks used for dimensionality reduction and data visualization

The variety of techniques invented in the field of ANN is very large, see [65] for a comprehensive review on existing ANN models. The particular algorithms listed above are among the most popular ones presently used as dimensionality reduction tools.

3.2.1 Self-Organizing Maps (SOM)

SOM [78] is probably the most popular ANN algorithm used for data visualization and is described in details in the next section 3.3.

3.2.2 Autoassociative Feedforward Neural Network (AFN)

Autoassociative Feedforward Neural Network [3] [84] allow dimensionality reduction by extracting the activity of d neurons of the internal “bottleneck” layer (containing fewer nodes than input or output layers) in an MLP. The network is trained to reproduce the data space, i.e. training data are presented to both input and output layers while obtaining a reduced representation in the inner layer.

3.2.3 Curvilinear Components Analysis (CCA)

Curvilinear Components Analysis [35] (by Vector Quantization and Projection) was proposed as an improvement to the Kohonen’s self-organizing maps, the output space of which is continuous and takes automatically the relevant shape. CCA is a neural network structured in two separate layers having D and d neurons respectively, and performing respectively vector quantization (VQ) and non-linear projection (P) from D-dimensional space to d-dimensional space. The weights of the first layer are the codebook of the vector quantizer. Vector quantization is performed by competitive learning, to which a regularization term (CLR) is added because a model of the distribution support is searched rather than the distribution itself. This regularization allows *unfolding* of data structures, that is dimension reduction of data lying on lines, surfaces or spheres embedded in higher-dimensional data spaces. The adaptation of the second layer’s weights is based on a minimization of a Sammon like measure:

$$E = \frac{1}{2}\sum_{i}\sum_{j\neq i}^{N}(D_{ij} - d_{ij})^2 F(d_{ij}, \lambda_y) \tag{3.1}$$

where $F(d_{ij}, \lambda_y)$ is a bounded and monotonically decreasing weighting function allowing the selection of the range of distances preferably preserved. This range is controlled by the radius λ_y generally evolving with the time, but it can also be controlled by the user, allowing an interactive selection of the scale at which the unfolding takes place. The minimization of E is achieved by a simplified (and fast) gradient-like rule. The speed-up of the algorithm is due to the fact that, at each iteration step, one point (randomly chosen) is pinned and all other points move around without regard to interactions amongst them. In this way the complexity of one minimization step scale (i.e. the number of inter-points distances to compute) only in N instead of N^2. This modification of the minimization process may explain the fact that CCA is little prone to get trapped in local minima as reported in [34]. CCA is also claimed to allow an *inverse projection*, that is from the 2-dimensional space to the D-dimensional space by a permutation of the input and output layers.

3.2.4 NeuroScale

NeuroScale [129] is a feed-forward neural network designed to effect a topographic, structure preserving, dimension-reducing transformation, with an additional facility to incorporate different degrees of associated subjective information. The implementation of this topographic transformation by a neural network is the following: A Radial Basis Function (RBF) neural network is utilized to predict the coordinates of the data points in the transformed data space. The weights of the network are adjusted in order to minimize the following error measure that embodies the topographic principle:

$$E = \sum_{i<j}^{N} (D_{ij} - d_{ij})^2 \tag{3.2}$$

where the D_{ij} are inter-point Euclidean distances in the data space and d_{ij} are the corresponding distances in the mapping space. If $\mathbf{x}_i = (x_1, ..., x_D)$ is an input vector mapped onto point $\mathbf{y}_i = (y_1, ..., y_d)$, we have $D_{ij} = ||\mathbf{x}_i - \mathbf{x}_j||$ and $d_{ij} = ||\mathbf{y}_i - \mathbf{y}_j||$. The points $\{\mathbf{y}_i\}$ are generated by the RBF, given the data points $\{\mathbf{x}_i\}$ as input. That is, if h is the number of neurons of the hidden layer and $\{\mathbf{\Phi}_i\}$ are the basis functions, $\mathbf{y}_i = \sum_{j=1}^{h} w_{ij} \mathbf{\Phi}_j(||\mathbf{x}_i - \mu_j||) = f(\mathbf{x}_i, \mathbf{W})$, where $f(\cdot, \mathbf{W})$ is the nonlinear transformation effected by the RBF with parameters (weights) $\mathbf{W}$. The error function (3.2) is expressed then as a function of the weights $\mathbf{W}$:

$$E = \sum_{i<j}^{N} (||\mathbf{x}_i - \mathbf{x}_j|| - ||f(\mathbf{x}_i, \mathbf{W}) - f(\mathbf{x}_j, \mathbf{W})||)^2 \tag{3.3}$$

that can be differentiated with respect to $\mathbf{W}$. Weight derivatives are calculated for *pairs* of input patterns and the network is trained for all the pairs of input patterns via any nonlinear optimization algorithm[1]. This scheme can include additional subjective knowledge concerning dissimilarity of each pair of data points, denoted s_{ij} (this knowledge can be for instance class information for generating data spaces that separate classes). This subjective knowledge is incorporated to the algorithm by replacing in equation (3.2) the data space distance D_{ij} with

$$\delta_{ij} = (1 - \alpha) \cdot D_{ij} + \alpha s_{ij}, \quad \alpha \in [0, 1] \tag{3.4}$$

where parameter α allows to control the degree of interpolation between purely geometric relationships and subjective knowledge.

[1] This training scheme is not supervised because we don't know a priori the positions $\mathbf{y}_i$, nor unsupervised because we know the relative distance for each pair of data, so it is called *relative supervision*.

3.2.5 Other neural network implementations of multidimensional scaling

The two methods previously described have the following common feature: they are implemented as neural networks, with a learning rule “borrowed” from multidimensional scaling in order to obtain topographic mappings. This idea was presented in other papers as Neural Network implementations of MDS [136] or as Neural Network for Sammon’s projection (SAMANN) [95] that implements an unsupervised backpropagation learning algorithm to train a multilayer feedforward neural network.

3.2.6 The Generative Topographic Mapping (GTM)

It must be noted first that this model developed in a statistical framework is not an artificial neural network, nevertheless it is described here because of its strong relation to the SOM model. The Generative Topographic Mapping [14] [123] is a probabilistic model that has been proposed as an alternative to the self-organizing maps in order to overcome the main difficulties encountered in the SOM model (see section 3.3.2). The GTM model is based on the assumption that the D observed variables are generated by L hidden or latent variables. The GTM defines a *generative* non-linear parametric mapping $\mathbf{y}(\mathbf{x},\mathbf{W})$ ($\mathbf{W}$ is a matrix of weights) from an L-dimensional latent or visualization space ($\mathbf{x} \in \Re^L$) to the D-dimensional data space ($\mathbf{y} \in \Re^D$) defined as

$$\mathbf{y}(\mathbf{x},\mathbf{W}) = \mathbf{W}\Phi(\mathbf{x}) \tag{3.5}$$

where the elements of $\Phi(\mathbf{x})$ consist of M fixed Gaussian basis functions. The probability distribution $p(\mathbf{x})$ over the latent space is defined in the form of a regular grid of K delta functions centered at the latent points $\{\mathbf{x}_k\}$ as

$$p(\mathbf{x}) = \frac{1}{K}\sum_k^K \delta(\mathbf{x}-\mathbf{x}_k) \tag{3.6}$$

since we do not expect the data to be confined exactly to the curved latent space, $p(\mathbf{x})$ is convolved with an isotropic Gaussian noise distribution given by

$$p(\mathbf{t}|\mathbf{x},\mathbf{W},\beta) = N\,(\mathbf{y}(\mathbf{x},\mathbf{W}),\beta) \tag{3.7}$$

where $\mathbf{t}$ is a point in the data space. Parameters $\mathbf{W}$ and β determine the mapping and they are estimated by a maximization of the log likelihood function ℓ:

$$\ell = \sum_n^N \log\left(\frac{1}{K}\sum_k^K p(\mathbf{t}_n|\mathbf{x}_k,\mathbf{W},\beta)\right) \tag{3.8}$$

through an Expectation-Maximization (EM) [36] procedure. In this way, the mapped latent distribution fits the observed data distribution. If the relationship between latent and observed variables is linear, this approach is known as *factor analysis* [91]. Using a non-linear mapping function, the distribution in the latent space will be non-linearly embedded in the data space on a curved manifold. The mapping is finally obtained using Bayes’ theorem in conjunction with the prior distribution over latent variable $p(\mathbf{x})$ to compute the corresponding posterior distribution in latent space for any given point $\mathbf{t}$ in data space as

$$p(\mathbf{x}_k|\mathbf{t}) = \frac{p(\mathbf{t}|\mathbf{x}_k,\mathbf{W}^\star,\beta^\star)p(\mathbf{x}_k)}{\sum_{k'} p(\mathbf{t}|\mathbf{x}_{k'},\mathbf{W}^\star,\beta^\star)p(\mathbf{x}_{k'})} \tag{3.9}$$

3.3 Kohonen's Self-Organizing Maps

3.3.1 Introduction

The algorithm called "Self-Organizing Map" introduced in 1981 by T. Kohonen [78] has been widely studied and applied over the past two decades (a list of more than 3000 references about SOM is given in [74]). SOM was originally devised by Kohonen to model a biological phenomena called *retinotopy*, a process of *self-organization* of the neural links between the visual cortex and the retina cells. The SOM is recognized as a gross simplification of this retinotopic process, which occurs in the brain. The SOM is a particular type of ANN that combines multivariate data visualization and clustering capabilities. One particularity of SOM is that the output layer of neurons is a two-dimensional array (map) that is directly used for data visualization purposes. A second feature of SOM is that the learning process is *unsupervised* or *self-organized*. This means that class information about the data is *not* used during the learning process, whether such information is available or not. The SOM neural network takes as input a set of high-dimensional sample vectors (labeled or not) and gives as output an array of codebook vectors, usually in one, two or three dimensions for visualization purposes. The basic two-layered structure of a SOM neural network is shown in figure 3.2. After training, the codebook can be used to display the training data or new data in a number of ways, as will be shown in section 3.3.4.

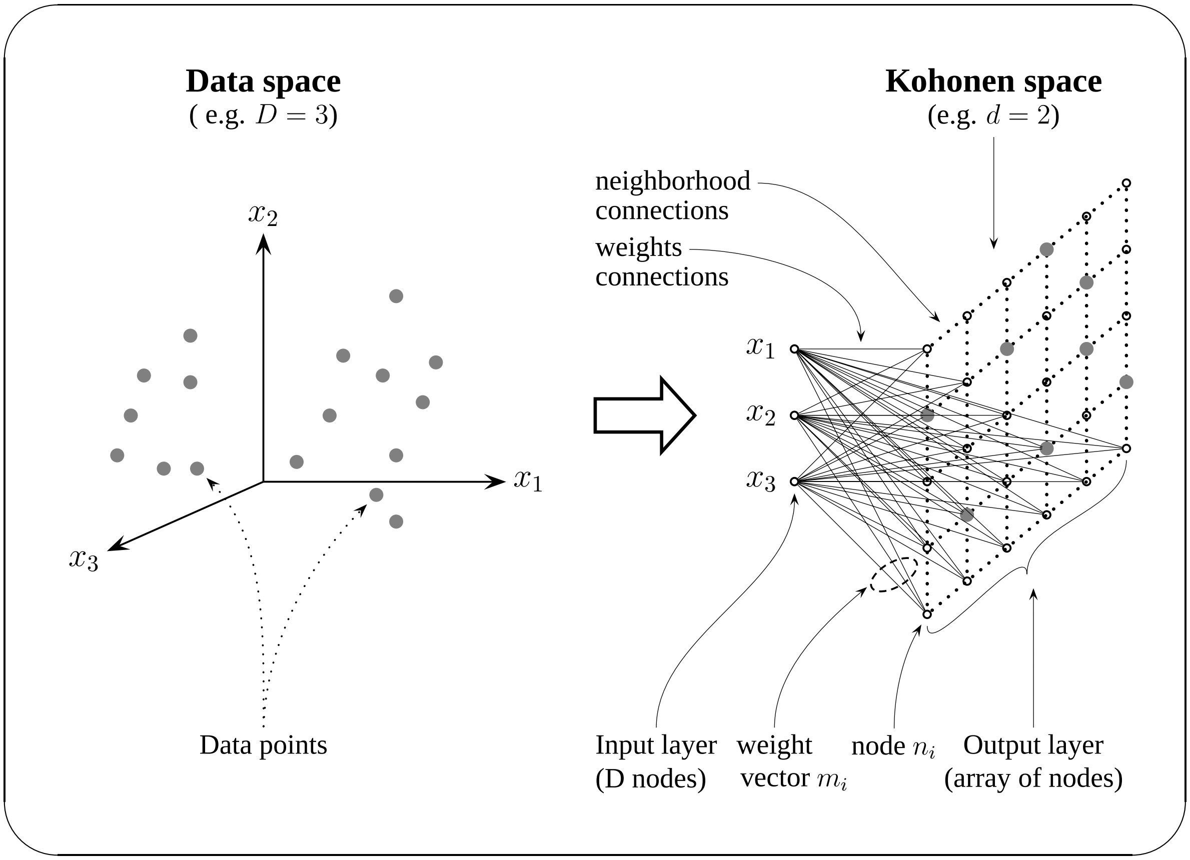


Figure 3.2: Data visualization using Self-Organizing Map Neural Network

Learning Vector Quantization

From the algorithmic point of view, SOM can be seen as an unsupervised version of a supervised-learning algorithm called Learning Vector Quantization (LVQ) [79] that was developed as a statistical classification tool. The idea of LVQ is that in a classification system (a classifier) based on the nearest-neighbor rule, a drastic gain of computation speed can be obtained reducing the number of vectors that represent each class. This reduction of the number of data vectors is a *clustering*. A set of reference vectors, also called *codebook vectors*, is adapted through an iterative process to the data according to a *competitive learning* rule. Competitive learning means that only the closest codebook vector (called winning or Best Matching Unit – BMU) is adapted (i.e. moved towards the presented data) at each iteration. This resembles a competition of the neurons to be activated. The training vectors as well as the reference vectors are categorical, i.e. labeled with a class name, and the training is qualified as *supervised* because it uses the class information of the training vectors. Let us note $x(t)$ the training vector presented at iteration t, $\{m_i\}$ the set of codebook vectors and $m_c(t)$ the nearest codebook vector to $x(t)$. Vector m_c is obtained from the equation:

$$\|x(t)-m_c(t)\| = min_j \|x(t)-m_j(t)\| \tag{3.10}$$

and is adapted according to the *learning rule*:

$$m_c(t+1) = \begin{cases} m_c(t)+\alpha(t)\cdot[x(t)-m_c(t)] & \text{if } C(x)=C(m_c), \\ m_c(t)-\alpha(t)\cdot[x(t)-m_c(t)] & \text{if } C(x)\neq C(m_c). \end{cases} \tag{3.11}$$

where $\alpha(t) \in [0,1]$ is a decreasing function of t called *learning rate* and $C(\cdot)$ is a function returning the class of a vector. It must be noted that only *one* unit (the BMU m_c) is adapted at each iteration, the codebook vectors of the remaining units are left unchanged. The result of the process is an approximation of the data probability density function by the codebook. After such a training of the codebook, a new vector y is classified according to the *nearest neighbor rule*: y is classified into class C_k if the nearest codebook vector to y is m_c, where $C(m_c) = C_k$.

Self-Organizing Maps principle

While in LVQ each unit is updated independently from the others, in SOM the unit interact in lateral directions because the codebook vectors are organized in a two-dimensional array and the learning rule has additional neighborhood constraints. The neighborhood constraints in SOM are that in the competitive learning rule, not only the winning unit is adapted, but also the units located in its neighborhood on the array. The underlying idea is that "*Global order can arise from local interactions.*" A *neighborhood function* h_{ci} that can be of type `"bubble"` or `"gaussian"`[2] is included into the LVQ learning rule (3.11), giving the following SOM learning rule applied to *all* codebook vectors:

$$m_i(t+1) = m_i(t) + h_{ci}(t)\cdot[x(t)-m_i(t)] \tag{3.12}$$

`*` If `"bubble"` neighborhood:
$$h_{ci}(t) = \begin{cases} \alpha(t) & \text{if } \|m_c-m_i\| \leq r(t) \\ 0 & \text{if } \|m_c-m_i\| > r(t) \end{cases} \tag{3.13a}$$

`*` If `"gaussian"` neighborhood:
$$h_{ci}(t) = \alpha(t)\cdot e^{\frac{-\|m_c-m_i\|}{2\sigma^2(t)}} \tag{3.13b}$$

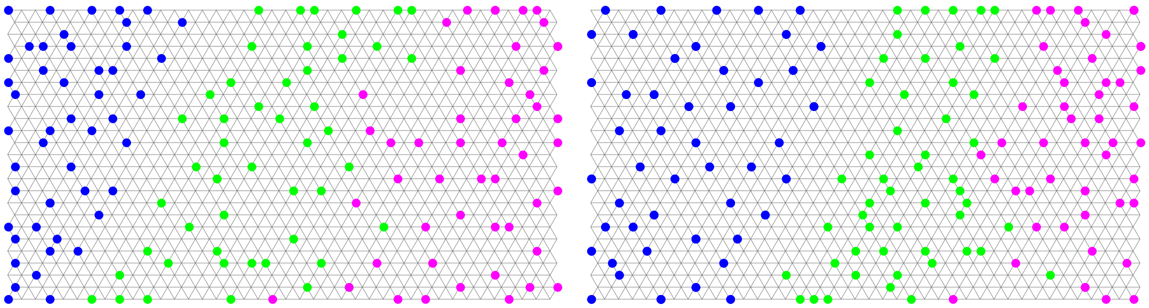
[2] In the `"bubble"` type, only one neuron is activated at each iteration, this is called *winner-takes-all* (WTA), whereas the `"gaussian"` type applies the *winner-takes-most* principle.

Only the neurons within the neighborhood $h_{ci}(t)$ around m_c are moved near to $x(t)$. $r(t)$ and $\sigma(t)$ are called neighborhood radiuses and are monotonically decreasing functions of t. This leads at the same time to a continuity of the codebook vectors (they are topologically ordered) over the array of units and to an approximation of the input data probability density function. These two features ensure that the resulting two-dimensional representation of the data tends to preserve the topography of the training data, which means that similar data will be mapped on neighboring areas of the map. From a statistical point of view, Kohonen self-organizing maps are discrete approximations to principal curves and surfaces.

3.3.2 Problems and limitations of the model

Varying areal magnification factors

If some areas of the data space are described by more data points than in the remaining data space, the corresponding area on the map will be large due to the tendency of the algorithm to minimize the quantization error functional. This effect is called the locally varying *magnification factor* [7]. The fact that the presentation of many similar data points to the network enhances their representation on the same area of the map can be interpreted as a clustering capacity of the algorithm. This clustering capacity cannot be practically exploited in classification tasks because of the simultaneous magnifications of different such areas on the map that reduce to a minimum the inter-cluster areas. This property is not desirable in data visualization tasks because local topology preservation then surpasses global topology preservation. The magnification effect has been illustrated in figure 3.3, where a SOM map was trained on the one hand using the original `iris`[3] data set 3.3(a), on the other hand using an augmented `iris` data set 3.3(b) in which vectors belonging to class `versicolor` occur twice. The area on the map coding class versicolor is trained using two times more training vectors and is hence more precisely coded, but this results in its magnification in figure 3.3(b) because the *number* of training vectors for this area is larger than in the case of the original data set, and not because the corresponding area in the data space is really larger. The same two data sets were also mapped using MDS, see §4.2.1 for a comparison.



(a) Map trained on the original `iris` data set

(b) Map trained on the augmented `iris` data set

Figure 3.3: The areal magnification effect: `iris` data set displayed on SOM maps ("hexagonal" topology, 40 × 25 nodes). The area coding class `versicolor` (blue dots) is magnified in 3.3(b).

[3] A famous data set widely used in the pattern recognition literature [47]. It contains three classes of 50 instances each, where each class refers to a type of iris plant (versicolor, setosa and virginica). The data is 4 dimensional, and consists of measurements of sepal and petal lengths.

Sensitivity to the initialization of the codebook

The final mapping of a data set strongly depends on the initialization of the codebook. If we want to get a good mapping, it is advised to train the codebook several times with different initial configurations and then to keep the “best” mapping. How much a given mapping is better than another one can be measured by the topology preservation measures that are presented in §3.3.3. In order to show the effect of a wrongly initialized map, we trained several times after random initialization the codebook of a square map with 20×20 neurons on the following data set: a number of points were randomly picked (under uniform distribution) from a 2-dimensional triangular space. The trained codebook was then displayed on the triangular surface to show how well it fits this space. It can be seen in figure 3.4 that the left map was initialized in a proper way which results in a quite uniform filling[4] of the data space by the map, whereas the right map wasn’t well initialized and ended up “twisted”.

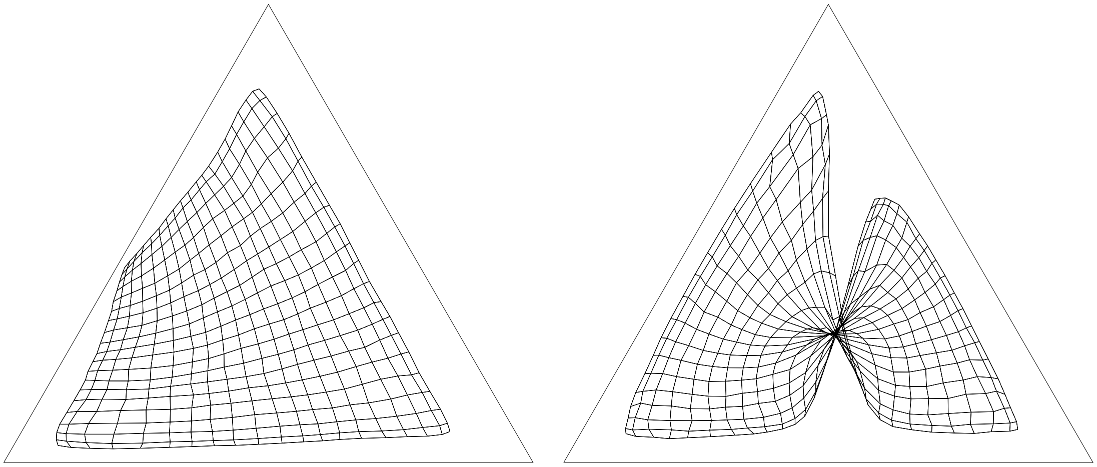


Figure 3.4: Codebook initialization: A square SOM network trained on the `triangle` data set (and displayed on the triangle surface) after two different random initializations.

Distortions connected with the map shape

Because the data tend to fill as much as possible the mapping space, the level of topology preservation highly depends on the fitting of the map shape to the data manifold shape. For instance an elongated manifold projected on a square map will be elongated in the direction perpendicular to its principal axis, tending to fit (and fill) the square. For this reason Kohonen suggests to first visualize the data using Sammon’s mapping to get an idea of the rough data manifold shape, and then design the network whose dimensions fit the data manifold. It has been also proposed to adapt the size of the map (by insertion or pruning of rows or columns of neurons) during learning in order to minimize distortion [9]. In order to visualize how the SOM array shape influences the resulting mapping, a `sphere` data set was mapped on a square SOM map. This artificially generated data consisted of 86 points on the surface of a sphere with radius 1 – 12 points on each of 7 equally spaced parallels, and the 2 poles. As shown in figure 3.5, this leads to distortions, especially on the corners of the array, due to shape differences between data and mapping spaces. Another effect of the map shape is the so called *border effect*, meaning that neurons at the border of the map are attracted towards the center (This can be seen in figure 3.4). To avoid these problems, Martinetz and Schulten proposed an algorithm called Neural Gas [96] close to Kohonen’s SOM, with the difference that the neurons of the

[4] The filling is not uniform in fact because we try to fit a square map into a triangle.

output layer are not arranged in a rectangular array of fixed shape, but they are rearranged at each iteration in a decreasing order of their distance to the currently presented training vector. Fritzke developed [51] a self-organizing network called Growing Cell Structure that adapts the number of neurons and its shape during learning (by adding or pruning only one neuron at a time) in order to better represent the data manifold.

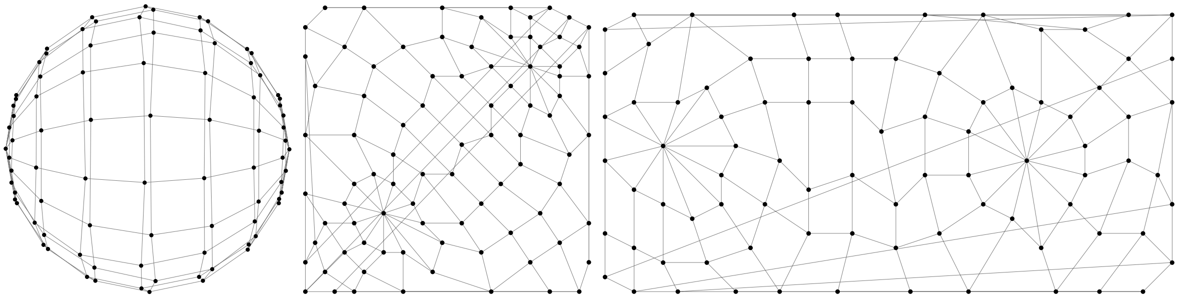

(a) Sammon's mapping (b) 30 × 30 nodes SOM (c) 40 × 20 nodes rectangular SOM

Figure 3.5: Distortions due to the map shape: A sphere mapped using two different SOM maps is visually much more distorted than using Sammon's mapping. The sphere is *unfolded* by SOM in a similar manner to the mappings obtained by CCA (see §3.2.3).

Discretization of the output space

The fact that the mapping is performed from a continuous data space onto a discretized space (the array of neurons of the output layer) is another restriction to a faithful representation of the data topography. An augmentation of the number of nodes on the map reduces of course this problem, at the cost of an increase of the training duration. This problem is especially crucial when the map is used to plot new data vectors in order to see its exact position on the map with respect to training data vectors. The discretized array of nodes offers a limited number of places where to plot the new point. In order to face this lack of continuity, Göppert et al. proposed [59] three techniques based on some interpolation in the output space:

- **Interpolation parameters by projection** This method consists of an orthogonal projection of the error vector (from the actual approximation to the exact input) onto the distance vector from the actual approximation to the next winner. For a given input vector $\mathbf{X}$, $w0$ denotes the index of the so called first winner, that is the closest codebook vector: $|\mathbf{X}-\mathbf{W}_{w0}| = min_i|\mathbf{X}-\mathbf{W}_i|$, and $\{wi, i = 1,...,k\}$ denote the indices of the further winning neurons that are the k topological neighbors of $\mathbf{W}_{w0}$ (on the grid of the output space). The following iterative procedure is repeated for all winners $(i = 1,...,k)$:

$$\tilde{\mathbf{X}}_0 = \mathbf{W}_{w0}^{(in)} \quad ; \quad \tilde{\mathbf{Y}}_0 = \mathbf{W}_{w0}^{(out)} \tag{3.14a}$$

$$\alpha_i = \frac{(\mathbf{X}-\tilde{\mathbf{X}}_{i-1})^T(\mathbf{W}_{wj}^{(in)}-\tilde{\mathbf{X}}_{i-1})}{(\mathbf{W}_{wj}^{(in)}-\tilde{\mathbf{X}}_{i-1})^T(\mathbf{W}_{wj}^{(in)}-\tilde{\mathbf{X}}_{i-1})} \tag{3.14b}$$

$$\tilde{\mathbf{X}}_i = \tilde{\mathbf{X}}_{i-1} + \alpha_i\left(\mathbf{W}_{wj}^{(in)} - \tilde{\mathbf{X}}_{i-1}\right) \tag{3.14c}$$

$$\tilde{\mathbf{Y}}_i^{(out)} = \tilde{\mathbf{Y}}_{i-1}^{(out)} + \alpha_i\left(\mathbf{W}_{wj}^{(out)} - \tilde{\mathbf{Y}}_{i-1}\right) \tag{3.14d}$$

- **Interpolation parameters by matrix inversion** In this method, we define a set of distance vectors $\{\mathbf{l}_i^{(in)}\}$ that form a local coordinate system $\mathbf{L}^{(in)}$:

$$\mathbf{l}_i^{(in)} = \mathbf{W}_{wi}^{(in)} - \mathbf{W}_{w0}^{(in)} \quad i = 1, \ldots, k \tag{3.15a}$$

$$\mathbf{X}^l = \mathbf{X} - \mathbf{W}_{w0}^{(in)} \tag{3.15b}$$

$$\mathbf{L}^{(in)} = [\mathbf{l}_1^{(in)} \mathbf{l}_2^{(in)} \cdots \mathbf{l}_k^{(in)}] \tag{3.15c}$$

  The local system in the output space $(\mathbf{L}^{(out)})$ is calculated accordingly. The base directions of the coordinate system are supposed to be linearly independent, but not orthogonal, so affine coordinates are obtained by the pseudo-inverse matrix $\mathbf{T}$:

$$\mathbf{T} = (\mathbf{L}^{(in)T}\mathbf{L}^{(in)})^{-1}\mathbf{L}^{(in)T} \tag{3.16a}$$

$$\alpha_i = \sum_{j=1}^{D} T_{ij} x_j^l \quad ; \quad \alpha_0 = 1 - \sum_{i=1}^{k} \alpha_i \quad i = 1, \ldots, k \tag{3.16b}$$

$$\mathbf{Y}^{(out)} = \mathbf{W}_{w0}^{(out)} + \mathbf{Y}^{l(out)} = \sum_{i=0}^{k} \alpha_i \mathbf{W}_{wi}^{(out)} \tag{3.16c}$$

- **Interpolation parameters by iterations** The first interpolation method does not lead to an optimal result because the distances vectors are not orthogonal. The second method achieves better results, but it is highly sensitive to noise. In this method, the iterative update rule is defined by the minimization of an error function by gradient descent:

$$E = \frac{1}{2}\sum_{j=1}^{D}(x_j^{(in)} - \tilde{x}_j^{(in)})^2 = \frac{1}{2}[\mathbf{X}^{l(in)} - \tilde{\mathbf{X}}^{l(in)}]^2 \tag{3.17a}$$

$$\Delta\alpha_i = \gamma \frac{(\mathbf{X}^{l(in)} - \tilde{\mathbf{X}}^{l(in)})^T \mathbf{l}_{wi}^{(in)}}{\mathbf{l}_{wi}^{(in)T}\mathbf{l}_{wi}^{(in)}} \quad i \in 1, \ldots, k \tag{3.17b}$$

  This procedure is inspired by the Delta rule [141].

Classification and clustering performance

SOM is an algorithm that performs a the same time vector quantization or clustering and visualization of high-dimensional data. Besides visualization of high-dimensional data, SOM has been applied to classification tasks. A codebook trained on categorical data constitutes a classifier by the use of the nearest neighbor rule applied to the codebook. It has been reported in [83] that SOM projection has a performance that is comparable or better than Sammon's mapping for the purpose of classification of clustered data. Many different classifiers have been compared on classical classification data sets in the framework of the StatLog project [98]. Compared to techniques that are only devoted to classification such as the k-Nearest Neighbors (KNN), Linear Discriminant Analysis (LDA) or other neural networks, the performances of SOM as a classifier are reported as poor. An attractive feature of SOM (that partly accounts for its popularity) is that the network learns the data in an unsupervised manner. The algorithm can therefore be used as a clustering pre-processing in unsupervised segmentation of textured images (to be used for example in medical image databases). The experiments we conducted in this direction [100] brought us to face the problem of the SOM map "segmentation", because SOM performs an unsupervised clustering but does not provide class definition. For this reason, SOM cannot be used alone for automatic unsupervised classification. SOM was not found to be a more efficient clustering tool than classical statistical clustering techniques such as the k–means.

Algorithmic aspects

Although one-dimensional Kohonen maps have been analyzed in some details little is known about the self-organization process in two or three dimensions [80]. The main problem is the lack of quantitative measure to determine what exactly “the good map” is. The problems listed here are quoted after [123]. The training algorithm does not optimize an objective function.[5] There is no general guarantee the training algorithm will converge. Convergence has been proven only under restricted conditions, e.g. the one-dimensional case [48] [128]. There is no theoretical framework based on which appropriate values for the model parameters can be chosen, e.g. initial value for the learning rate and width of the neighborhood functions, and subsequent rate of decrease and shrinkage, respectively. Some rules were provided in this matter in [99] and are based on an stochastic approximation theory approach. Those problems make difficult the use of the algorithm, requiring from the user knowledge about appropriate parameters that should be used.

3.3.3 Data topography preservation and its measures

In order to clarify the purpose of the measures presented in this section, we will first introduce the following two definitions from the McGrawHill Encyclopedia of Science and Technology, vol.13, pp.667-683:

- **Topology** “The study of topological spaces and continuous maps”. The important point is the continuity of the data space in neighborhoods based on a metric.
- **Topographic surveying and mapping** “The measurement of surface features and configuration of an area or region, and the graphic expression of those features”. The purpose here is to represent a structure or configuration of objects.

In the remain of this text, we are interested in the relative positions of points in the data space revealing the structure of data manifolds, so we want to measure the preservation of its topography, but the terms topology and topography preservation are used quite interchangeably. Data represented in a D-dimensional space need not be really D-dimensional, e.g. points picked-up from a plane that is embedded in a 3-dimensional space. The effective dimensionality of a data manifold is denoted here D_i and called *intrinsic dimensionality*. D_i is necessarily smaller than the number of non zero eigenvalues (i.e. the rank of the data matrix $\mathbf{X}$, and its value can be estimated in a number of different manners [34] [122], (the number of the first eigenvalues that are significantly larger than the remaining ones is a good first guess). The embedding of a D-dimensional manifold in a d-dimensional map space ($d \ll D$) leads to more or less local distortions of data topography. These data topography distortions are related to the reduction of dimensionality, and they will be more important when the difference $D_i - d$ increase. The measures presented in this section were designed to provide numerical indicators of how much a given mapping leads to a better data topography preservation than another mapping. Such indicators allow to compare different mappings obtained by different SOM networks and to retain the best one. The best mapping is the one for which a measure of the topology distortion induced by the mapping is the smallest. The main difficulty to design such a measure is that SOM output space is not a continuous space as the input space is, but an array of nodes.

[5] It has been proved [45] that such an objective function cannot exist in the general case of a continuous distribution function of the input data. But in the case of finite pattern manifolds (that is in most of practical applications), the input data set is finite and the SOM does have a (local) objective function which is the reconstruction error expressed by equation (3.19) [73] [113]. SOM learning rule corresponds to a gradient descent step towards a stochastic approximation of the minimum of equation (3.19).

SOM learning quality measures

Given that very different learning processes can be defined starting with different initial codebook vectors, and applying different learning parameters, it can be useful to define an indicator of the quality of a given SOM training round. Such an indicator, used by Kohonen, is the average square quantization error defined as

$$q_e^1 = \frac{1}{T} \sum_{t=1}^{T} \|x_t - m_c\| \tag{3.18}$$

or the average distortion measure

$$q_e^2 = \frac{1}{T} \sum_{t=1}^{T} \sum_{i=1}^{N} h_{ci} \|x_t - m_i\|^2 \tag{3.19}$$

where h_{ci} is the neighborhood function defined in (3.12), N is the number of nodes in the array and T is the number of training vectors. In fact these measures only reflect how well the codebook matches the training data, measuring the quality of the learning process but giving no indication about the data topography preservation of the mapping.

SOM mapping quality measures

It is not obvious to define precisely the intuitive notion that a mapping preserves the "internal structure" of the data in one space in another space. Two possible choices are to say that the mapping must preserve some distance measures, or that the mapping must preserve distances orderings. Bauer and Pawelzik defined [8] a *topographic product* as a measure of preservation or violation of neighborhood relations as follows:

$$P = \frac{1}{N(N-1)} \sum_{j=1}^{N} \sum_{k=1}^{N-1} \log(P_3(j,k)) \tag{3.20a}$$

$$P_3(j,k) = \left(\prod_{l=1}^{k} \frac{d^V(w_j, w_{n_l^A(j)})}{d^V(w_j, w_{n_l^V(j)})} \frac{d^A(j, n_l^A(j))}{d^A(j, n_l^V(j))} \right)^{\frac{1}{2k}} \tag{3.20b}$$

P is an average of $P_3(j,k)$ over all neighborhood orders k and neurons j, $n_k^A(j)$ denote the k-th nearest neighbor of node j, with the distances $d^A(j,k)$ measured in the output space and in the same way $n_k^V(j)$ denote the k-th nearest neighbor of node j, with the distances $d^V(w_j, w_k)$ measured in the input space. A vanishing value of P indicates a perfect neighborhood preservation; negative (positive) values indicate a too small (too large) output space dimensionality. Another approach to data topography preservation measure for a SOM mapping was proposed in [83] and consists in computing Sammon error measure after having previously defined some distances in the input and output spaces. Bezdek and Pal argue [11] that these methods do not measure topology preservation directly, and they introduce a more general measure called *Metric Topology Preserving index* as:

$$\rho_{S_p} = 1 - \frac{6 \sum_{k=1}^{T} (r^\star(k) - r(k))^2}{T^3 - T} \tag{3.21}$$

where index k runs over the $T = n(n-1)/2$ distances and $r^\star(k)$ (respectively. $r(k)$) is the rank of distance $d^\star(k)$ (resp. $d(k)$) in the input (resp. output) space. ρ_{S_p} is a Spearman's rank correlation coefficient indicating distances order preservation. [58] gives a summary of different definitions of what a "perfectly neighborhood preserving" map is and presents some of their properties.

Information preservation measure

The following information loss measure can be defined [39]:

$$A(\mathbf{Y}) = \frac{\sum\limits_{i>j}^{N} (d_{ij}(\mathbf{Y}) - \delta_{ij})^2}{\sum\limits_{i>j}^{N} d_{ij}(\mathbf{Y})^2 + \sum\limits_{i>j}^{N} \delta_{ij}^2} \tag{3.22}$$

where $A(\mathbf{Y}) \in [0,1]$ indicates how much of the information contained in the data inter-points distances is preserved by the mapping.

3.3.4 Applications of SOM to multivariate data visualization

Many different ways to use SOM as a data visualization tool have been proposed, depending on the type of data to visualize and on the purpose of the visualization. The methods listed here below use the capacity of the SOM mapping to display the topographic relationships of the data vectors used to train the map. See [137] for a more complete review of SOM-based data visualization methods. Other techniques allow for example to see the evolution of temporal data in the form of a trajectory on the trained map.

- A first simple solution is to put the label of the learning individuals at the place of their BMU, providing a display of the training data vectors as shown in figure 3.3. This visualization technique can be applied if the data are class-labeled and their number is smaller than the number of neurons on the map. Even then the main limitation is that very often several data vectors fall on the same map neuron, hence a problem to visualize such data.

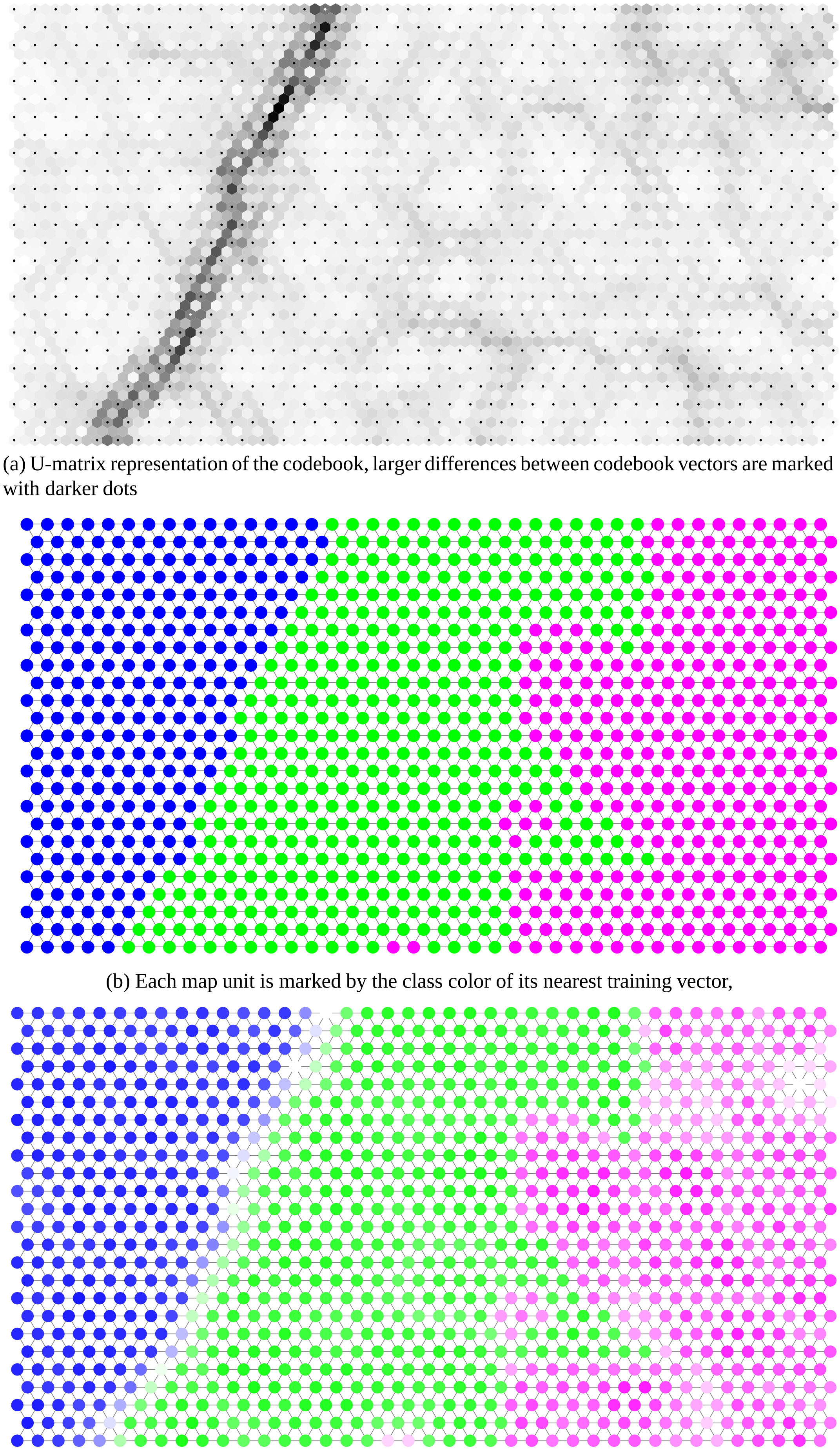
- In the case where data are not labeled, it is possible to mark the place of each unit with the number of training data vectors for which this unit is the BMU, marking the BMU whether with the number itself, or using a gray-level color proportional to this number. This gives a map containing frequency information (a kind of 2-dimensional histogram), hence revealing clusters. But those representations do not show the local distances between the adjacent units, giving no idea of the real data space distances existing between the possible clusters on the map.

- Several authors [80], [83], [26] chose to put into light the clusters using shades in a gray scale depending on the mean distance in the codebook vectors space of a unit to its direct neighbors. One such method is the unified distance matrix (U-matrix) method [135] [71], in which the gray value of each pixel is determined by the maximum distance in the data space of the corresponding unit to its four neighbors in the network (smaller distances are white whereas larger distances are black). Map areas where the codebook vectors are significantly varying code larger parts of the data space than areas with more constant codebook vectors, thereby the U-matrix gives an estimation of areal magnification factors. An example of U-matrix display (obtained using the program `umat` of the SOM_PAK package [81] distribution) is given in figure 3.6(a).

- Another possibility to visualize categorical data is to color the map by class clusters. This can be done by marking each unit of the map with a colored disc, each color corresponding to the one class of the training data for which this unit is BMU. In this way each class is represented by a colored area on the map, as shown in figure 3.6(b). Here again there can be conflicting data, i.e. data from different classes for which the same unit is

BMU. Additionally, the class color labelling each map unit can be tuned or “contrasted” proportionally to the average quantization error of the unit, with null saturation (white color) on the unit with the largest quantization error and fully saturated color on the unit with smallest quantization error. Another possibility is to let the label radius instead of color saturation depend on quantization error. In this way, information concerning the confidence we can have in the class position and shape is reported on the map by colors intensities, as can be seen in figure 3.6(c).

- The WEBSOM is an extension of SOM that is widely used for data visualization of economical analyses [33] [73], as well as in data mining of texts or Knowledge Discovery in Databases (KDD) [69]. The idea of the WEBSOM method lies in its two-level architecture, where the upper level map takes in input the histogram of the lower level map. In the case of data mining of texts, a category map of words (lower level) is built on the basis of appearances of words in a thesaurus, giving words-meaning proximities that are used to build a document map (upper level). The visualized document map provides a general view of the document collection.



(a) U-matrix representation of the codebook, larger differences between codebook vectors are marked with darker dots

(b) Each map unit is marked by the class color of its nearest training vector,

(c) As on figure above 3.6(b) with quantization error based contrast added.

Figure 3.6: Different visualizations of `iris` data set trained on a SOM map with “hexagonal” neighborhood of 40 × 25 nodes.

Chapter 4

The statistical approach: Multidimensional scaling

4.1 Introduction

4.1.1 Narrow and broad sense definitions of multidimensional scaling

The term multidimensional scaling (MDS) is commonly employed with two quite different meanings [32]:

- In the first meaning (the narrow sense), MDS is a family of the techniques aimed at representing some dissimilarity data in a low dimensional space. A definition close to this one of multidimensional scaling is given in [27, p. 1]:

  > Suppose a set of *N* objects is under consideration and between each pair of objects (i, j) there is a measurement δ_{ij} of the "dissimilarity" between the two objects. A narrow definition of MDS is the search for low dimensional space, usually Euclidean, in which points in the space represent the objects, one point representing one object, and such that the distances between the points in the space $\{d_{ij}\}$ match as well as possible the original dissimilarities $\{\delta_{ij}\}$.

- In the second meaning (the broad sense), MDS is said to cover any technique which produces a graphical representation of objects from multivariate data. Such a definition includes various forms of cluster analysis or statistical multivariate analysis methods, such as PCA or Correspondence Analysis (CA) [10] [63].

  > A wider definition of multidimensional scaling can subsume several techniques of multivariate data analysis. At the extreme it covers any technique which produces a graphical representation of objects from multivariate data. [27, pp. 1-2]

  Ripley follows the same approach in his book [112, pp. 305-311] defining MDS as a family of techniques including all the methods producing a two-dimensional display of objects on the basis of their similarities or distances, such as Sammon's Non Linear Mapping [115] or Principal Components Analysis.

In the following we will focus on MDS in the narrow sense, and on the simplest case of *one-mode two-way* data. This terminology means that the observed objects are known only by one dissimilarity measurement (one-mode) per couple of objects (two-way). Data dimensionality reduction using MDS consists therefore in the following two steps:

STEP 1: Compute inter objects distances in the data space $\mathcal{D}$,

STEP 2: Input those distances as dissimilarities to the MDS algorithm.

The choice for the distance measure is usually the Euclidean distance (and this choice was made in the entirety of this work). This choice is not the only one possible, for example higher order Minkowski distances (with $k > 2$) or Malahanobis distances may be more appropriate in some cases. Dissimilarities involving more than two objects could also be considered (triplets, etc.). The choice of Euclidean distance was made here for the sake of simplicity, but in real applications it should be driven by the nature of the data. Particularly in the case where some of the data attributes are not numerical and continuous, but just binary or symbolic, a carefully taylored distance function should be envisaged.

4.1.2 Overview of Multidimensional Scaling techniques

Origins of the method

The collection of methods referred to as multidimensional scaling can be regarded as a nonlinear graphical technique. These methods were originally designed to deal with data collected directly in the form of inter-point distances (or distances-like quantities) between pairs of objects. Multidimensional scaling (MDS) is a set of data analysis techniques that display the structure of distance-like data as a geometrical picture. MDS has its origins in psychometrics, where it was proposed to help understand people's judgements of the similarity of members of a set of objects, e.g. perceived distances between colors, olfaction substances, etc. The first computer programs implementing MDS were developed by Shepard, Kruskal, Young, Guttman and Lingoes, finding their foundations in the theoretical works of Torgerson [132] and Coombs [25]. Torgerson proposed the first MDS method and coined the term. MDS has now become a general data analysis technique used in a wide variety of fields.
The basic idea of MDS consists in finding through iterations a configuration of points representing as well as possible in a two-dimensional space a set of compared objects. If inter objects similarity measures are not directly available, they can be obtained by computation of some distance in the high dimensional space. It must be noted that geometric Euclidean distance may not always reflect correctly the significant similarities between the data vectors (especially when handling with nominal or ordinal discrete attributes, or with very different scalings). The inter-point distances in the produced configuration should fit as well as possible the given similarities (the different implementations of the fitting will be described later). Hence, the MDS algorithm takes as input numerical values describing similarities between N points, arranged in a $[N \times N]$ similarity matrix, and produces an array of two-dimensional coordinates of the representing points, the configuration matrix. Let us note that MDS do not put any hypothesis or restriction on the data distribution, nor on the data space or its metric. The dissimilarities are not necessarily distances, so they don't need to fulfill neither the triangle inequality of Euclidean metrics $d_{ac} \leq d_{ab} + d_{bc}$ nor the symmetrycity condition $d_{ab} = d_{ba}$. These properties are to be related to the fact that MDS methods were initiated in psychometric studies, in which dissimilarity measurements were given by subjects quantifying perceived closeness between objects in rank scales (e.g. not at all, a few, very much, ...). The application of MDS only requires the possibility to sort the dissimilarities in increasing or decreasing order.

Domains of application of MDS

A classical application of MDS is the re-construction of a map containing cities of a given country on the basis of journey times between the cities. Other applications can be found in

[12], for example the construction of a space of persons working on a given job, of a space of similar companies, or in studies of consumption goods. [22] list numerous scientific fields in which MDS was applied, e.g. in marketing studies [37], in econometrics [93], in political sciences [140] and in sociology [18], [28]. The methods was most successfully applied in mathematical psychology, psychometrics and geography [130], [106], [55].

Particular approaches and recent developments

- Statistical inference:
  - A statistical approach to MDS presented by Ramsay [111] consists in the modelling of dissimilarities incorporating an error function, which leads onto inferential procedures. The idea is to let the observed dissimilarity between objects i and j, conditioned on d_{ij}, have probability density function $p(\delta_{ij}|d_{ij})$. It is assumed that these conditioned observations are independent and identically distributed and hence the log-likelihood is $\ell = \sum_r \sum_s \ln p(\delta_{ij}|d_{ij})$. The distances can be written in terms of the coordinates of the points, $d_{ij}^2 = (x_i - x_j)^T(x_i - x_j)$, and hence the log-likelihood can be minimized with respect to x_i and any parameters of the probability density function p. This gives the maximum likelihood estimates of the coordinates, $\hat{x}_i$. Two possible distributions for $\delta_{ij}|d_{ij}$ are the normal and log-normal. Another similar maximum likelihood method as been proposed by Takane [127] for nonmetric scaling.
  - A different probabilistic approach is the one of Tsuchiya [133] in which not the inter-points distances but the points coordinates are modelled by a normal distribution. Means and variances are estimated by a maximum likelihood procedure using the EM algorithm.
- Global optimization methods:
  - The *tunneling method* was applied by Groenen to perform global minimization of the Stress [64]. The tunneling method alternates a local search step, in which a local minimum is sought, with a tunneling step in which a different configuration is sought with the same Stress as the previous local minimum. In this manner successively better local minima are obtained. The crucial tunneling step is realized by defining a tunneling function that will be zeroed by iterative majorization. Suppose $\mathbf{Y}^\star$ is a local minimum configuration, the tunneling function is defined as:

    $$\tau(\mathbf{Y}) = (S(\mathbf{Y}) - S(\mathbf{Y}^\star))^{2\lambda} \left(1 + \frac{1}{\sum\limits_{ij}(d_{ij}(\mathbf{Y}) - d_{ij}(\mathbf{Y}^\star))^2}\right) \tag{4.1}$$

    $\tau(\mathbf{Y})$ has zero points for configurations with Stress equal to $S(\mathbf{Y}^\star)$. The reader is referred to Groenen for details on how these zero points can be found.
  - In the case of unidimensional scaling, Pliner [109] proposed to face the problem of the great number of local minima by smoothing the function to minimize. Although not guaranteed inevitably to locate the global optimum, the smoothing technique did so in all computational experiments where the global optimum was known. The smoothing is achieved by integrating the Stress function as follows:

    $$S_\varepsilon(\mathbf{x}) = (1/\varepsilon^n) \int\limits_{D(\mathbf{x},\varepsilon)} S(\mathbf{y})\, d\mathbf{y} \tag{4.2}$$

where $D(\mathbf{x}, \varepsilon)$ is a cube in $\Re^n$ with the center in $\mathbf{x}$ and a side ε. The integrated Stress function $S_\varepsilon(\mathbf{x})$ is then minimized by a gradient descent method.

– Klock and Buhmann introduced a *deterministic annealing* algorithm for the SStress function (see §4.4.1, equation (4.16)) and for Sammon mapping, derived in the framework of maximum entropy estimation [77]. Deterministic annealing has been designed to gather the advantages of both deterministic search techniques (such as gradient descent) and stochastic simulated annealing, that is both rapidity and efficiency. The idea is to replace the sampling of the Gibbs distribution of the coordinates by exact or approximated calculations of the relevant expectation values w.r.t. the Gibbs distribution.

4.1.3 Metric and non-metric MDS

In general the dissimilarities are obtained from experiments, e.g. a subject's judgement of similarity between each possible pairs from a given set of objects. In order to facilitate the comparison of those dissimilarities with distances, the dissimilarities will first be transformed so as to make them closer to distances, after which a configuration of points will be iteratively sought so that the distances match as well as possible the transformed dissimilarities. If the transformation applied to the dissimilarities preserves the metric properties of distances then it is *metric MDS*, if not it is *non-metric MDS*. If the similarities (or dissimilarities) are proportional to distances, the ensuing method is called metric MDS. If the dissimilarities are assumed to be merely ordinal, it is non-metric MDS. In the case of metric MDS, an analytic expression of the coordinates can be obtained by a procedure called *classical scaling* described below. In the case of non-metric MDS, a configuration is sought so that distances $\{d_{ij}\}$ between pairs of points in the space match "as well as possible" the original dissimilarities $\{\delta_{ij}\}$. This is usually performed through numerical minimization of a loss function (or goodness of fit function) expressing how well the inter-point distances fit the given dissimilarities.

- Metric least-squares scaling is based on the minimization of a loss function with respect to the coordinates of the d-dimensional points. Methods belonging to this category are Sammon's non-linear mapping (NLM) [115] (it is a metric scaling because dissimilarities are Euclidean distances, but not linear because there does not exist a matrix transforming the input points into the projected ones), ALSCAL [126], an alternating least squares scaling method and SMACOF [31], a minimization method using a majorizing function. ALSCAL and SMACOF are alternatives to the gradient methods for the minimization of Stress.

- Nonmetric scaling is also least-squares, but only the rank order of the dissimilarities is taken into account, hence the nonmetric character of the method.

4.1.4 Classical scaling

Torgerson [131] used a theorem proved by Young and Householder [142] which shows that starting with a matrix of Euclidean distances, it is possible to determine the dimensionality of the Euclidean space, and the coordinates of the points in this space for which inter-points distances are exactly the given distances. Classical scaling procedure can be summarized as follows:

1. Let the searched coordinates of N points in a D dimensional Euclidean space be given by $\mathbf{x}_i \quad (i = 1, \ldots, N)$, where $\mathbf{x}_i = (x_{i1}, \ldots, x_{iD})^T$. Matrix $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]^T$ is the $N \times D$

coordinates matrix. The Euclidean distances $\{d_{ij} = (\mathbf{x_i} - \mathbf{x_j})^T(\mathbf{x_i} - \mathbf{x_j})\}$ are known. The inner product of matrix $\mathbf{X}$ is denoted $\mathbf{B} = \mathbf{X}\mathbf{X}^T$. Find matrix $\mathbf{B}$ from the known distances $\{d_{ij}\}$ using Young-Householder process [142]:

   (a) Define matrix $\mathbf{A} = [a_{ij}]$ where $a_{ij} = -\frac{1}{2}d_{ij}{}^2$,

   (b) Deduce matrix $\mathbf{B}$ from $\mathbf{B} = \mathbf{HAH}$, where $\mathbf{H} = \mathbf{I} - \frac{1}{N}\mathbf{1}\mathbf{1}^T$ is the centering matrix,

2. Recover the coordinates matrix $\mathbf{X}$ from $\mathbf{B}$ using the spectral decomposition of $\mathbf{B}$:

   (a) The inner product matrix $\mathbf{B}$ is expressed as $\mathbf{B} = \mathbf{X}\mathbf{X}^T$. The rank of $\mathbf{B}$ is $r(\mathbf{B}) = r(\mathbf{X}\mathbf{X}^T) = r(\mathbf{X}) = D$. $\mathbf{B}$ is symmetric, positive semi-definite and of rank D, and hence has D non-negative eigenvalues and $N - D$ zero eigenvalues.

   (b) Matrix $\mathbf{B}$ is now written in terms of its spectral decomposition, $\mathbf{B} = \mathbf{V}\Lambda\mathbf{V}^T$, where $\Lambda = diag(\lambda_1, \lambda_2, ..., \lambda_N)$ the diagonal matrix of eigenvalues $\{\lambda_i\}$ of $\mathbf{B}$, and $\mathbf{V} = [\mathbf{v}_1, ..., \mathbf{v}_N]$ the matrix of corresponding eigenvectors, normalized such that $\mathbf{v}_i^T\mathbf{v}_i = \mathbf{1}$,

   (c) Because of the $N - D$ zero eigenvalues, $\mathbf{B}$ can now be rewritten as $\mathbf{B} = \mathbf{V}_1\Lambda_1\mathbf{V}_1^T$, where $\Lambda_1 = diag(\lambda_1, \lambda_2, ..., \lambda_D)$ and $\mathbf{V}_1 = [\mathbf{v}_1, ..., \mathbf{v}_D]$,

   (d) Finally the coordinates matrix is given by $\mathbf{X} = \mathbf{V}_1\Lambda_1^{\frac{1}{2}}$, where $\Lambda_1^{\frac{1}{2}} = diag(\lambda_1^{\frac{1}{2}}, ..., \lambda_D^{\frac{1}{2}})$

If we extract the $d \quad (d < D)$ first principal coordinates of the spectral decomposition of $\mathbf{B}$, then we get a Principal Coordinates Analysis (PCO) [60]. [1] When dissimilarities are used instead of Euclidean distances to define matrix A and then to produce matrix $\mathbf{B}$, it is interesting to ask under what circumstances $\mathbf{B}$ can give rise to a configuration of points in Euclidean space. The answer is that if $\mathbf{B}$ is positive semi-definite of rank p, then a configuration in p dimensional Euclidean space can be found [32]. In this case, classical scaling approximates the set of dissimilarities with distances corresponding to a configuration of points in a Euclidean space. To summarize, we can say that Classical Scaling (or Principal Co-ordinates Analysis) is essentially an algebraic method of reconstructing point coordinates assuming that the dissimilarities are Euclidean distances. In dimensionality reduction tasks, the objects coordinates in D dimensional space are given, so classical scaling (and Principal Coordinates Analysis – PCO) amounts to a principal components analysis – PCA of the given coordinates.

4.2 Least Square Scaling algorithm

4.2.1 The Stress function

Least Square Scaling is based on the minimization of a function called loss function, badness or goodness-of-fit function, error function or Stress function (we will use the last name in the text below). This scalar measures how well the representation configuration matches the given dissimilarities, the lower is its value, the better is the match. A general expression of the Stress function is as follows:

$$S(\mathbf{Y}) = \frac{1}{F_n}\sum_{i<j}^{N_t} w_{ij} \cdot \left(\delta_{ij} - d_{ij}(\mathbf{Y})\right)^2 \tag{4.3}$$

[1] PCA is based on a spectral decomposition of a correlation matrix $\mathbf{S} = k \cdot \mathbf{X}^T \cdot \mathbf{X}$, whereas PCO is based on a spectral decomposition of a scalar product matrix $\mathbf{B} = k \cdot \mathbf{X} \cdot \mathbf{X}^T$. It is well known that the matrices $\mathbf{X}^T \cdot \mathbf{X}$ and $\mathbf{X} \cdot \mathbf{X}^T$ have the same non zero eigenvalues (but different eigenvectors) [57], so the two methods, based on linear transformations, lead to similar results. Moreover the left and right singular vectors of X are particular choices of the eigenvectors of $\mathbf{X} \cdot \mathbf{X}^T$ and $\mathbf{X}^T \cdot \mathbf{X}$, respectively[49, p. 203].

where (i, j) are indices running over the mapped points, $i < j$ means $i = 1, \ldots, N_t - 1$ and $j = i+1, \ldots, N_t$, δ_{ij} is a dissimilarity between objects O_i and O_j that can be a given input data, or a computed geometric distance separating objects O_i and O_j in a high-dimensional data space $\mathcal{D}$, or a pseudo-distance (sometimes called "disparity") derived from dissimilarities by a matching of rank orders, d_{ij} is a computed distance separating points P_i and P_j in the low-dimensional mapping space $\mathcal{M}$, $\{w_{ij}\}$ are factors introduced to weight the dissimilarities $\{\delta_{ij}\}$ individually and F_n is a normalization factor designed to make the Stress value independent to configuration scalings (shrinking or stretching).

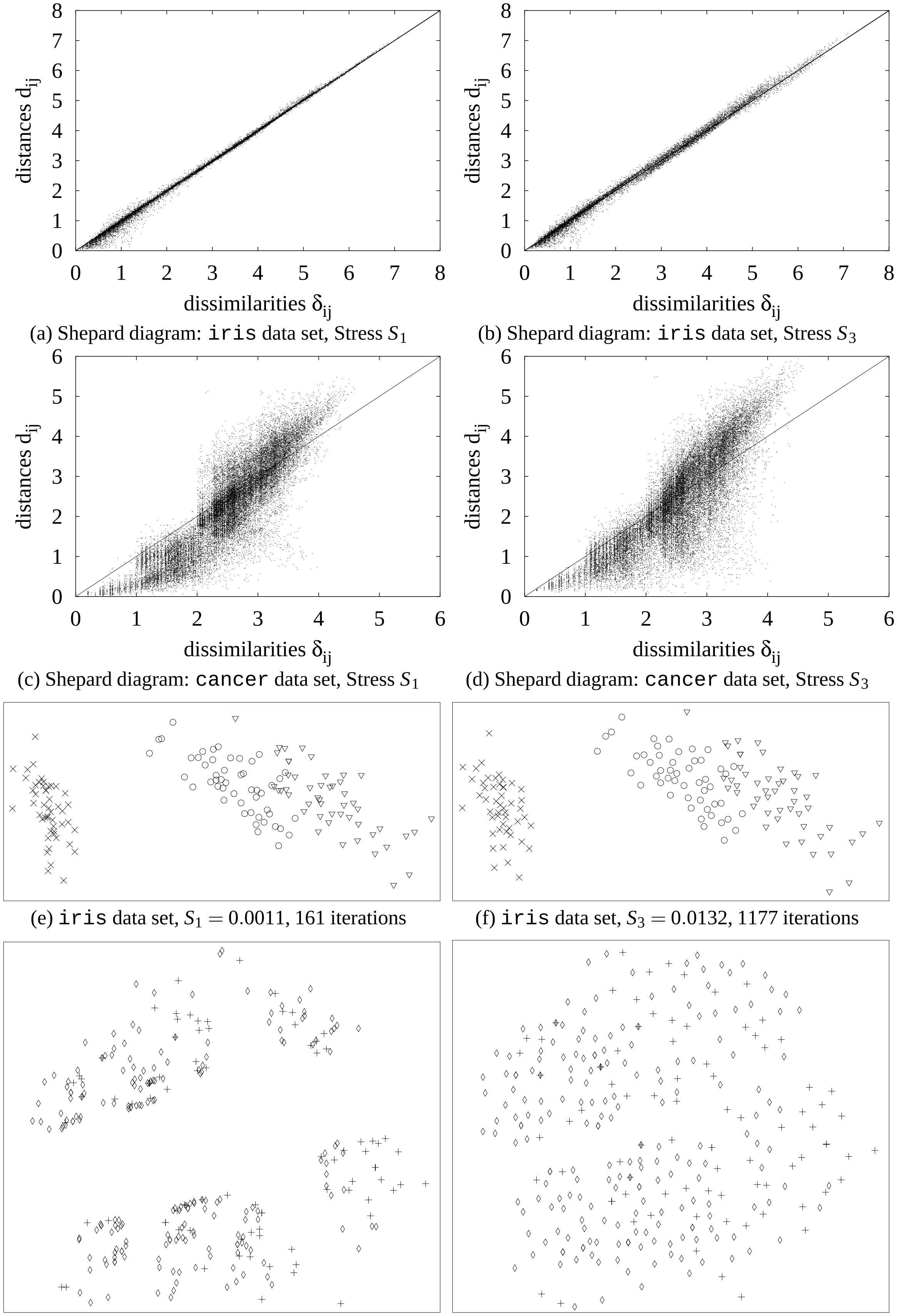
The question of how to choose the proper Stress function for a given task has been discussed in [88] and [43]. Depending on the Stress function chosen, the resulting configuration will be such that some inter-point distances are better preserved than others, e.g. smaller or larger distances. Following Duda and Hart [43, pp. 243-244], we can define three particular Stress expressions derived from the general expression (4.3) using the following weights and normalization factors:

$$w_{ij} = 1 \qquad F_n = \sum_{i<j}^{N_t} {\delta_{ij}}^2 \qquad S_1(\mathbf{Y}) = \frac{1}{\sum\limits_{i<j}^{N_t} {\delta_{ij}}^2} \cdot \sum_{i<j}^{N_t} \left(\delta_{ij} - d_{ij}(\mathbf{Y})\right)^2 \tag{4.4}$$

$$w_{ij} = \frac{1}{\delta_{ij}} \qquad F_n = \sum_{i<j}^{N_t} \delta_{ij} \qquad S_2(\mathbf{Y}) = \frac{1}{\sum\limits_{i<j}^{N_t} \delta_{ij}} \cdot \sum_{i<j}^{N_t} \frac{\left(\delta_{ij} - d_{ij}(\mathbf{Y})\right)^2}{\delta_{ij}} \tag{4.5}$$

$$w_{ij} = \frac{1}{{\delta_{ij}}^2} \qquad F_n = N_t(N_t - 1) \qquad S_3(\mathbf{Y}) = \frac{1}{N_t(N_t - 1)} \cdot \sum_{i<j}^{N_t} \left(\frac{\delta_{ij} - d_{ij}(\mathbf{Y})}{\delta_{ij}}\right)^2 \tag{4.6}$$

In Stress expression of equation (4.4), each error $\delta_{ij} - d_{ij}$ is added directly to the sum regardless whether it is large or small, corresponding to an *absolute* error (this is Kruskal's choice, see §4.2.4, equation (4.13), with the minor difference that F_n sums over the output distances $\{d_{ij}\}$ instead of the dissimilarities). In equation (4.5), the squared errors are divided by δ_{ij}, corresponding to an *intermediate* error (Sammon's choice, see §4.2.3, equation (4.7)). Finally, equation (4.6) corresponds to an *relative* error because each error $|\delta_{ij} - d_{ij}|$ is weighted by $1/\delta_{ij}$. Consequently the participation into the sum of larger dissimilarities δ_{ij} with respect to smaller dissimilarities will be reduced in Stress S_3 compared to the sum in S_1. At the same time, the weights of small dissimilarities will increase, leading finally to their better preservation through Stress minimization. This last observation is confirmed by experiments on real data using the following diagrams.



(a) Shepard diagram: `iris` data set, Stress S_1

(b) Shepard diagram: `iris` data set, Stress S_3

(c) Shepard diagram: `cancer` data set, Stress S_1

(d) Shepard diagram: `cancer` data set, Stress S_3

(e) `iris` data set, $S_1 = 0.0011$, 161 iterations

(f) `iris` data set, $S_3 = 0.0132$, 1177 iterations

(g) `cancer` data set, $S_1 = 0.061$, 208 iterations

(h) `cancer` data set, $S_3 = 0.084$, 1054 iterations

Figure 4.1: MDS mappings (PCA initialization) obtained for Stress functions S_1 and S_3, and the corresponding Shepard diagrams for `iris` and `cancer` data sets.

The scatter plot of the dissimilarities $\{\delta_{ij}\}$ versus the distances $\{d_{ij}\}$, called *Shepard diagram* in the MDS literature, shows which distances better reflect their given dissimilarities. Points on this plot should roughly approach the diagonal line $y = x$. The closer the points to the diagonal, the better is the preservation of the dissimilarities. Figures 4.1(a) to 4.1(d) represent Shepard diagrams obtained after minimization of Stress functions S_1 and S_3 for two data sets : `iris` [2] and `cancer` [3]. We observe that minimizing Stress S_1 leads to a better preservation of larger distances than smaller ones for both data sets, because in figure 4.1(a) the diagonal line is thiner in its top-right end, and the cloud is closer to the diagonal line in figure 4.1(c). On the other hand, minimizing Stress S_3 preserves worse larger distances and better smaller ones than S_1. This is confirmed for both data sets: In figure 4.1(b), the right end of the diagonal cloud is thicker than in figure 4.1(a), and in figure 4.1(d), the left end of the cloud is closer to the diagonal line than in figure 4.1(a) whereas the right part of the cloud is further from the diagonal. We can notice that the smaller distances don't seem to be much better preserved by S_3 than by S_1 for the `iris` data set, because the left end of the diagonal is still thick in figure 4.1(b). This is due to the fact that in `iris` data set there are much more smaller distances than larger ones, as shown on histograms in figure 4.2.

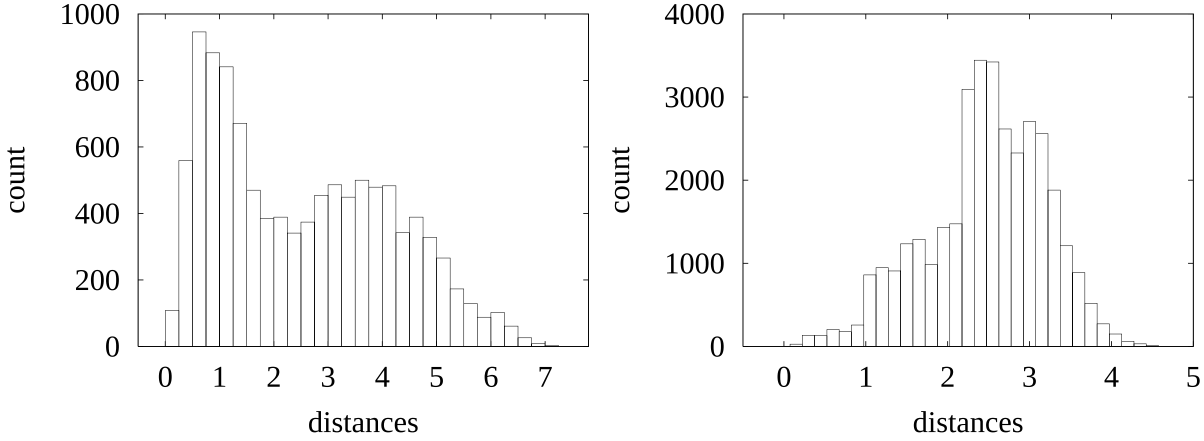


Figure 4.2: Histograms of inter-points distances d_{ij} for `iris` (left) and `cancer` (right).

As the Stress function is a sum over all the distances, it will be more difficult to change significantly the distances that are in greater number in the histogram, so for `iris` data set larger distances are much easier to change than smaller ones. Smaller distances are easier to chance in `cancer` data set because they do not form a peak on the histogram. The resulting configurations shown in figures 4.1(e) to 4.1(h) confirm this assessment: figures 4.1(e) and 4.1(f) are almost identical, whereas figures 4.1(g) and 4.1(h) differ importantly. Thus we put forward the following conclusion: The choice of the Stress weights expressions can importantly influence the resulting mapping, depending on the input data set, and more precisely on the histogram of its inter-points distances. The differences should be more important for data sets with histogram containing a peak in the middle than for data sets with histogram in which a peak is on one extreme. This assertion, based on the two previous data sets, would need more experimental verifying.

Another interesting effect of the difference between Stresses S_1 and S_3 can be observed

[2] The `iris` data set and its augmented version, used to illustrate the magnification effect in SOM (fig. 3.3), were mapped using Stress functions S_1 and S_3 (with random moves within spheres of radius $r = 0.01$ for identical data). Final configurations are almost identical, showing that the magnification effect does not occur in MDS mappings.

[3] Breast cancer from the UCI repository, obtained from the University Medical Centre, Institute of Oncology, Ljubljana, Yugoslavia.

during the mapping process that presents roughly two stages: at the beginning the configuration quickly stabilises, then a fine tuning stage takes place. When a data set is mapped using S_1, the fine tuning stage consists in movings of individual points, whereas in the case of S_3, it consists in movings of clusters of global structures.

4.2.2 Outline of the algorithm

The algorithm described in this part is generic for most of the existing computer implementations of Least Squares Scaling. The minimization of the Stress function $S(\mathbf{Y})$ is generally achieved by a gradient descent technique which is a *local* minimization, so it is prone to get stuck in local minima. As will be shown in a further section, the Stress function has many local minima, hence the crucial importance of the choice for the initial configuration: this configuration is generally defined whether randomly or by a Principal Components Analysis of the data points. In the most general case, $S(\mathbf{Y})$ sums over all the pairs of mapped data points, so the computation of $S(\mathbf{Y})$ and $\nabla S(\mathbf{Y})$ have a complexity of order N^2. As a consequence the number N of mapped points is a crucial parameter for the algorithm and the technique is prohibitive for large data sets. An outline of the MDS algorithm is given here below:

```
// Iterative MDS algorithm:

Define an initial configuration $Y^{(0)}$ [4]

while ($S^{(t)}$ has not converged) do

        1. normalize current configuration $Y^{(t)}$ [5]
        2. compute distances $\{d_{ij}\}$ for current configuration $Y^{(t)}$
        3. compute target distances $\{\hat{d_{ij}}\}$ (by monotonic or polynomial regression) [6]
        4. compute Stress value $S^{(t)}$
        5. compute gradient $\nabla S^{(t)}$
        6. compute step-size $\alpha^{(t)}$
        7. compute new configuration $Y^{(t+1)} = Y^{(t)} - \alpha^{(t)} \nabla S^{(t)}$

end while
```

The convergence criterion used to stop the iterations can be one or a combination of the following criterions:
- a given number *Nb_It* of iterations were performed,
- the current Stress value $S^{(t)}$ is smaller than a threshold value S_{min},
- the relative Stress decrease $2(S^{(t)} - S^{(t-1)})/(S^{(t)} + S^{(t-1)})$ between two consecutive iterations is smaller than a threshold value ε_S,
- the Stress gradient length $G^{(t)} = \|\nabla S^{(t)}\|$ is smaller than a threshold value G_{min}.
The numerous experiments made on various data sets showed that the best choice is to use a combination of the first and the third criterions, with for example $Nb\_It = 1000$ and $\varepsilon_S = 10^{-12}$. This allows to go as far as possible in Stress minimization convergence, with a limit to avoid minimization processes that never ends.

[4] whether randomly, by a PCA mapping or any other heuristic.

[5] this step is necessary to avoid configuration stretching indefinitely. It is not necessary if the normalization factor F_n sums output distances $\{d_{ij}\}$ instead of dissimilarities $\{\delta_{ij}\}$ as in equation (**??**).

[6] this step is present in non-metric MDS only, see §4.2.4 - section 'Computation of target distances'.

4.2.3 Sammon's non-linear mapping

Sammon introduced a technique called Non-Linear Mapping (NLM) [115] for the analysis of multivariate data. The algorithm is based on a point mapping of D-dimensional vectors from the data space $\mathcal{D}$ to a lower-dimensional such that "the inherent structure of the data is approximately preserved under the mapping". The distance between vectors with indices i and j in the D-dimensional space is the Euclidean distance and is denoted D_{ij}. Sammon defines an error E as

$$E = \frac{1}{\sum_{i<j}^{N} D_{ij}} \sum_{i<j}^{N} \frac{(D_{ij} - d_{ij})^2}{D_{ij}} \tag{4.7}$$

which is minimized using the following rule:

$$y_{pq}{}^{(t+1)} = y_{pq}{}^{(t)} - (MF) \cdot \frac{\frac{\partial E^{(t)}}{\partial y_{pq}{}^{(t)}}}{\left| \frac{\partial^2 E^{(t)}}{\partial y_{pq}{}^{(t)2}} \right|} \tag{4.8}$$

where MF is the so called "magic factor" which value was determined empirically to be optimal when $MF \in [0.3, 0.4]$. It can be seen that expression (4.8) is not a steepest descent rule, but a Newton method's (see equation 4.25) in which the Hessian matrix is approximated by its diagonal. In a correspondence [86], Kruskal points out that Sammon's algorithm is a particular type of his multidimensional scaling, computed by the program M-D-SCAL (version 5M) that minimizes the expression:

$$\frac{\textit{Sammon's error } E}{\sum \frac{d_{ij}^2}{D_{ij}^2}} \tag{4.9}$$

The denominator is connected with the fact that in Kruskal's program, the normalizing factor F_n sums over the output distances $\{d_{ij}\}$, whereas in Sammon's mapping F_n sums over the input distances $\{D_{ij}\}$. Kruskal argues that "the denominator under Sammon's error is so nearly constant over the region of interest that it hardly changes the resulting configuration". This is true if, as Kruskal does, the configuration is rescaled at each iteration to prevent it from growing.

4.2.4 Kruskal's non-metric scaling

The original program implementing Kruskal's nonmetric scaling is called KYST from the initials of its authors [90]. This technique originating in psychometrics differs from previous one in its nonmetric character, which comes from the fact that only the rank order of the dissimilarities (and not their magnitudes) is considered during the mapping process. The searched configuration must be such that the inter-points distances $\{d_{ij}\}$ satisfy as much as possible the relationships:

$$\forall i, j, k, l: \qquad \delta_{ij} \leq \delta_{kl} \quad \Rightarrow \quad d_{ij} \leq d_{kl} \tag{4.10}$$

The situation when two dissimilarities are equal is called a *tie* and the decision about how to constraint the corresponding distances is discussed in the paragraph "Treatment of ties".

Construction of the Stress function

Kruskal motivates his choice for the Stress function by the fact the criterion that will be minimized should measure how well the inter-point distances match the given data dissimilarities. The idea is to "make a scatter diagram of distances versus dissimilarities and, since a direct relationship is expected between these two quantities, to do a regression and measure how bad the fit is by using the residual sum of squares from the regression." The simplest kind of regression is a straight line of slope 1 through the origin. In this case, the regression sum of squares called *raw stress* and noted $S^\star$ reduces to:

$$S^\star = \sum_{i \neq j}^{N} (d_{ij} - \hat{d}_{ij})^2 \tag{4.11}$$

where (i,j) are indices running over the N mapped points, $i \neq j$ means $i = 1, ..., N$ and $j = 1, ..., i-1, i+1, ..., N$. [7] Then a scale or a normalization factor is defined in order to make the Stress invariant to shrinking of the configuration of points (many other scale factors are possible):

$$F_n = \sum_{i \neq j}^{N} d_{ij}^2 \tag{4.12}$$

Finally, the square root of the normalized stress is taken in order to improve interpretability (by giving a standard deviation-like measure), which leads to the following expression for *Stress S*:

$$S = \sqrt{\frac{\sum_{i \neq j}^{N} (d_{ij} - \hat{d}_{ij})^2}{\sum_{i \neq j}^{N} d_{ij}^2}} \tag{4.13}$$

Computation of the target distances

The calculation of the so called target distances $\{\hat{d}_{ij}\}$ is performed in Kruskal's original MDS using *monotone regression*, also called ordinal or isotonic regression [4]. Another procedure applied by Shepard in this purpose is the *rank-image permutation* [118] that was related to his particular Stress function. The monotone regression procedure allows to compute the target distances $\{\hat{d}_{ij}\}$ on the basis of the inter points distances $\{d_{ij}\}$ of the current configuration $Y(t)$ in such a way that the rank order of the $\{\hat{d}_{ij}\}$ matches as much as possible the rank order of the given dissimilarities $\{\delta_{ij}\}$. This procedure is illustrated in figure 4.3, where we can see that the target distances $\{\hat{d}_{ij}\}$, derived from the inter-point distances $\{d_{ij}\}$, have the same rank order as the dissimilarities $\{\delta_{ij}\}$.

[7] In nonmetric MDS, dissimilarities are often given as input data measurements that need not be symmetric, which explains that the sum over the points indices must be for all $i \neq j$ in this section. In the following sections, we will be concerned with symmetric similarities, since they are derived from computation of distances in the data space $\mathcal{D}$, then a summation for all $i < j$ is sufficient.

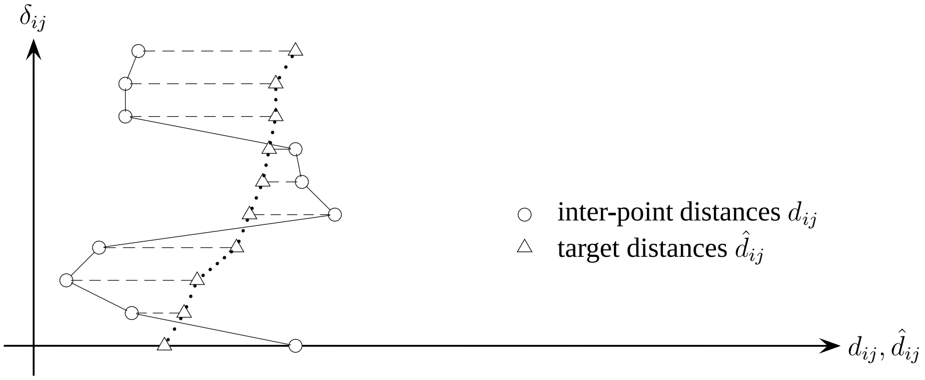


Figure 4.3: Shepard diagram illustrating the monotone regression procedure.

Here follows an outline of the algorithm of the monotone regression procedure :
This procedure takes as input the dissimilarity matrix rank orders $\{\delta_{ij}\}$ and the actual configuration distances $\{d_{ij}\}$, and outputs a set of target distances $\{\hat{d}_{ij}\}$. First sort the dissimilarities $\{\delta_{ij}\}$ in increasing order, and reorder the inter-point distances $\{d_{ij}\}$ correspondingly. Then draw a scatter plot of the distances (or Shepard diagram), in which each point $P_k, k = 1,...,N(N-1)/2$ has coordinates (d_{ij}, δ_{ij}). Link the points in the order of increasing dissimilarities. If the curve thus obtained is monotonically increasing, then the inter-point distances and dissimilarities are ordered in the same way and the inter-point distances are target distances. If not, we have to find such target distances for which the curve will grow monotonically with the dissimilarities, and that are as close as possible to the inter-point distances in the least squares sense. In brief, monotone regression consists of working consecutively through the distance values, checking whether they are in the same order as the given dissimilarities. When an inversion appears, i.e. when one or more distance values decreases, then a 'block' is formed by taking the offending value and the preceding one. These are averaged until monotonicity is restored between blocks. The sequence consisting of repeatedly comparing the target distances and the dissimilarities orders is illustrated in steps 1 through 5 of Table 4.1. This table reported from [29, p. 52] shows on an example the successive steps of a monotone regression.

Dissimilarities (δ_{ij})											
	Object n°		1	2	3	4	5				
		1	–								
		2	3	–							
		3	6	4	–						
		4	10	5	7	–					
		5	2	8	9	1	–				

		(5,4)	(5,1)	(2,1)	(3,2)	(4,2)	(3,1)	(4,3)	(5,2)	(5,3)	(4,1)
Objects indices (column, row)		(5,4)	(5,1)	(2,1)	(3,2)	(4,2)	(3,1)	(4,3)	(5,2)	(5,3)	(4,1)
Dissimilarity sorted into order (δ_{ij})		1	2	3	4	5	6	7	8	9	10
Distances in *current* configuration (d_{ij})		3	6	3	5	8	10	13	11	9	15
Calculation of target distances:											
Monotone regression	step 1	3									
	step 2		4.5	4.5							
	step 3				5	8	10				
	step 4							12	12		
	step 5							11	11	11	15
Target distances $(\hat{d}_{ij})$		3	4.5	4.5	5	8	10	11	11	11	15

Table 4.1: Derivation of target distances using Kruskal's monotone regression procedure.

The fact that such a procedure is inserted in the optimization process of the Stress function does not allow the complete optimization of the Stress function in the same way as for Sammon's mapping, because at each iteration the target distances are different. We will see in our experiments that this handicap is of little practical importance.

Treatment of ties

Suppose there are dissimilarities which by chance are precisely equal to one another. How then to interpret the constraint of the monotone relationship of the $\{\hat{d}_{ij}\}$ to the $\{\delta_{ij}\}$? The problem is to decide how to sort the corresponding target distances. Kruskal provides two solutions to this problem. One, which he calls the *primary approach* because it seems preferable, is to say that when $\delta_{ij} = \delta_{kl}$ we do not care which of d_{ij} and d_{kl} is larger nor whether they are equal or not, so we do not constrain $\hat{d}_{ij}$ and $\hat{d}_{kl}$. The *secondary approach* is to say that $\delta_{ij} = \delta_{kl}$ is evidence that d_{ij} ought to equal d_{kl}, so to impose the constraint $\hat{d}_{ij} = \hat{d}_{kl}$. The first approach was chosen for all experiments reported in this work.

Computation of the step-size

At the time Kruskal developed his program, most of the gradient descent methods let the step-size be proportional to the magnitude of the gradient (that is, they had a constant value of α). Sammon in his nonlinear mapping designed a "Magic Factor" determined by experience to a value of 0.3 or 0.4. Kruskal introduced an innovation by making the step-size be dependent on the angle between the present gradient and the gradient of the previous step. Kruskal's rationale [87, p. 316] for his step-size procedure is as follows:

> "After we have taken one step, we can see whether the step we have just taken is too large or too small by looking at the angle between the gradient where we are now and the gradient along which we have just moved. If the new gradient points in almost the same direction as the old one (so the angle between them is near 0°), then we should have taken a larger step. If the new gradient is perpendicular to the old one, the step we just took was about right. If the new gradient points back almost in the direction we came from (so the angle is near 180°), then the step we just took was too large."

Kruskal recommended for the calculation of $\alpha^{(t)}$ [87, pp. 318-319]:

$$\alpha^{(t+1)} = \alpha^{(t)} \times \mathit{angle\_factor} \times \mathit{good\_luck\_factor} \times \mathit{bias\_factor} \tag{4.14a}$$

where:

$$\begin{aligned}
\mathit{good\_luck\_factor} &= \sqrt{\min(1, \mathit{stress\_ratio}^{(t)})} && (4.14\text{b})\\
\mathit{angle\_factor} &= 4^{(\cos\theta)^3} && (4.14\text{c})\\
\mathit{bias\_factor} &= \frac{1.6}{[1+\mathit{av\_stress\_ratio}^{(t)^5}][1+\mathit{av}|\cos\theta|^{(t)} - |\mathit{av}\cos\theta|^{(t)}]} && (4.14\text{d})\\
\mathit{stress\_ratio}^{(t)} &= \frac{S^{(t+1)}}{S^{(t)}} && (4.14\text{e})\\
\mathit{av\_stress\_ratio}^{(t)} &= \mathit{stress\_ratio}^{(t-1)^{\frac{1}{3}}} \times \mathit{av\_stress\_ratio}^{(t-1)^{\frac{2}{3}}} && (4.14\text{f})\\
\theta &= (\widehat{\nabla S^{(t+1)}, \nabla S^{(t)}}) && (4.14\text{g})\\
\mathit{av}|\cos\theta|^{(t+1)} &= \frac{2}{3}|\cos\theta| + \frac{1}{3}\mathit{av}|\cos\theta|^{(t)} && (4.14\text{h})\\
|\mathit{av}\cos\theta|^{(t+1)} &= \frac{2}{3}\cos\theta + \frac{1}{3}|\mathit{av}\cos\theta|^{(t)} && (4.14\text{i})
\end{aligned}$$

4.2.5 A comparison of metric and non-metric MDS

We are now concerned with the comparison of metric and non-metric algorithms from an experimental point of view. We want to find out how those two Stress minimization processes differ, trying to answer the questions: “What does the monotone regression give more ?” and “Which of the two methods leads to the best representation ?” It comes to mind just to compare the two final Stress values (the lowest being the best), under the conditions: Start with the same initial configuration, use the same Stress function and the same stopping criterion. This amounts to taking for the non-metric mapping algorithm an exact copy of the used metric mapping algorithm, in which data dissimilarities $\{\delta_{ij}\}$ are replaced by disparities $\{\hat{d_{ij}}\}$ and a monotone regression step as in section “Computation of target distances” in previous §4.2.4 is added. It appears that, due to this monotone regression, a rescaling of the configuration is performed at each iteration and the final configuration is stretched compared to the one obtained from the metric algorithm[8] (because we use a normalisation factor depending on the $\{\delta_{ij}\}$) and, for this reason, a direct comparison of the two Stress values is meaningless. In order to enable such a comparison, one must whether: use a Stress function invariant to configuration rescaling (that is with $F_n = f(\{d_{ij}\})$), or one of the two final configurations (obtained from the metric and non-metric MDS algorithms) should be rescaled so that it fits as well as possible the other one. A particular field of multidimensional scaling called *Procrustes analysis* is precisely devoted to such a matter, addressing the more general question: “How to rescale, rotate, reflect and translate a configuration of points in order to make it match as well as possible to another configuration of points, under the condition of a one-to-one correspondence of the points ?” A short review of Procrustes analysis is given in [27, pp. 92-104]. Although this method was designed to match to one another configurations of data in two different spaces, we can use it to match the two final configurations obtained from metric and non-metric MDS.

[8] This property seems to be common to many iterative procedures. It can be related to neural networks learning by the Hebbian rule where the weights have to be normalized to prevent them to grow without bounds [53, p. 36].

Procrustes analysis

Let us denote $\mathbf{Y_m}$ (respectively $\mathbf{Y_{\overline{m}}}$) the matrix of the final configuration obtained from the metric (resp. non-metric) MDS algorithm, with $\mathbf{y}_{m,i}$ (resp. $\mathbf{y}_{\overline{m},i}$) the vector representing point P_i. The purpose of Procrustes analysis is to minimize the following expression:

$$R^2 = \sum_{i=1}^{N} (\mathbf{y}_{m,i} - \mathbf{y}_{\overline{m},i})^T (\mathbf{y}_{m,i} - \mathbf{y}_{\overline{m},i}) \qquad (4.15)$$

In the purpose of the visual comparison of metric and non-metric MDS final configurations, we decided to rescale the non-metric MDS final configuration so as to make it as close as possible to the metric MDS final configuration. This was achieved by the Procrustes analysis technique, summarized as follows:

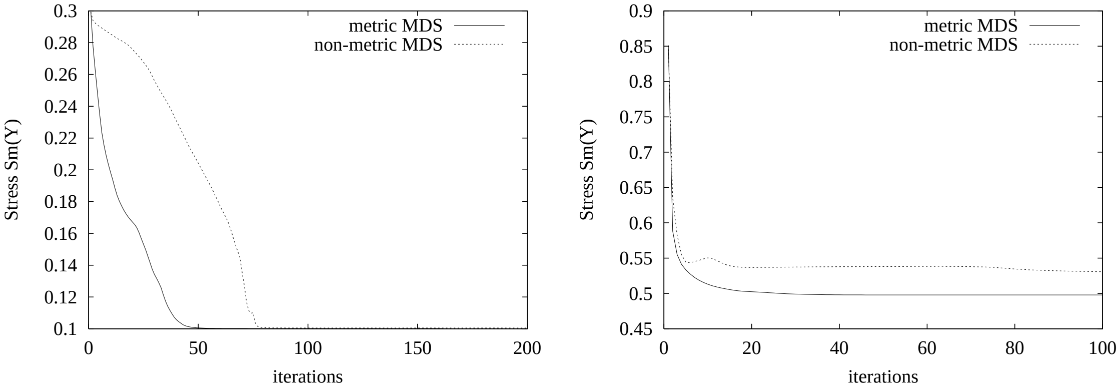
- Center configurations $\mathbf{Y_m}$ and $\mathbf{Y_{\overline{m}}}$ by subtracting their mean vectors column-wise,
- Find the rotation matrix $\mathbf{A} = (\mathbf{Y_{\overline{m}}}^T \mathbf{Y_m} \mathbf{Y_m}^T \mathbf{Y_{\overline{m}}})^{\frac{1}{2}} (\mathbf{Y_m}^T \mathbf{Y_{\overline{m}}})^{-1}$ and rotate the $\mathbf{Y_{\overline{m}}}$ configuration to $\mathbf{Y_{\overline{m}} A}$,
- Scale the $\mathbf{Y_{\overline{m}}}$ configuration by multiplying each coordinate by ρ, where $\rho = tr(\mathbf{Y_{\overline{m}}}^T \mathbf{Y_m} \mathbf{Y_m}^T \mathbf{Y_{\overline{m}}})^{\frac{1}{2}} / tr(\mathbf{Y_{\overline{m}}}^T \mathbf{Y_{\overline{m}}})$

See [27, pp. 92-104] for more detailed analytical derivations of those expressions.

Comparison of Stress values

Let us denote $S_m(\mathbf{Y})$ the Stress function minimized in the metric MDS algorithm, and respectively $S_{\overline{m}}(\mathbf{Y})$ in non-metric MDS. Because the two functions are different (target distances $\{\hat{d}_{ij}\}$ in $S_{\overline{m}}(\mathbf{Y})$ replace dissimilarities $\{\delta_{ij}\}$ in $S_m(\mathbf{Y})$), a direct comparison of those values is meaningless. We decide to compare the two methods using the metric MDS Stress expression $S_m(\mathbf{Y})$ for both configurations, computing the values $S_m(\mathbf{Y_m})$ and $S_m(\mathbf{Y_{\overline{m}}})$. The comparisons were based on two small[9] data sets. One of them, called the `animals` data set, borrowed from [82], is made of descriptions of 16 animals using 13 binary features (is small, median, big, has 2 legs, 4 legs, hair, hooves, mane, feathers, likes to hunt, run, fly, swim). The second data set called `cmyk` is made of 68 colors with for attributes (their c, m, y and k components) expressed as real numbers between 0 and 1. Convergence curves for both data sets are shown in figure 4.4, where we can see that the final Stress values are very close for the `cmyk` data set, and quite different for the `animals` data set.

[9] because the program used for the Procrustes analysis (from the diskette accompanying [27]) takes maximally in input 100 data points.



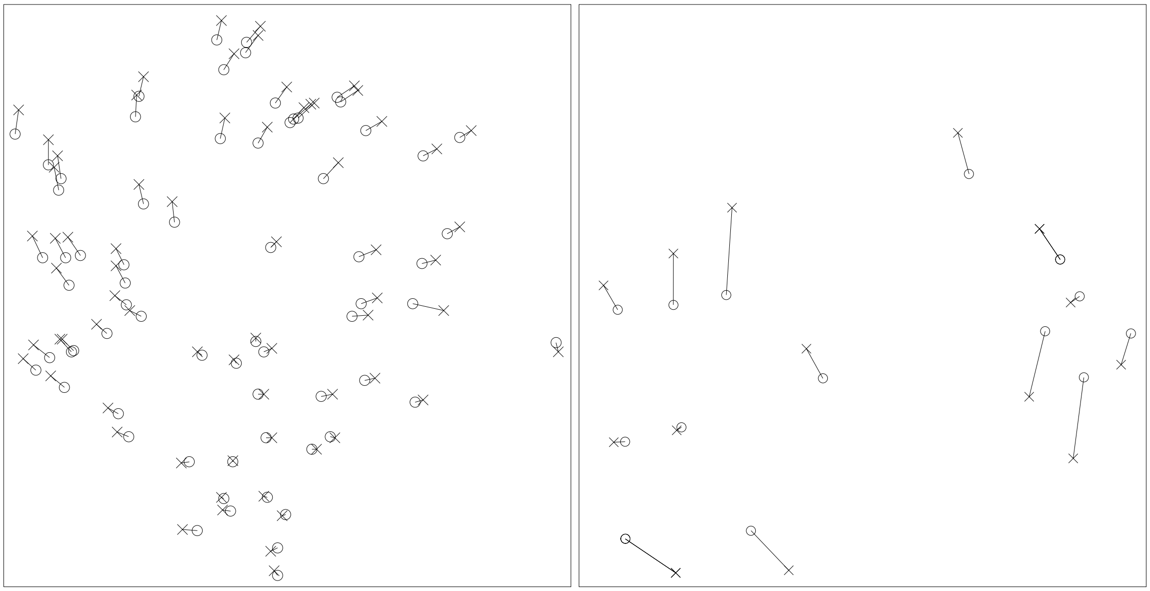
(a) `cmyk` data set.

(b) `animals` data set.

Figure 4.4: Comparison of metric and non-metric MDS minimization processes.

Visual comparison of the resulting configurations

In order to visualize the differences in points positions between the final configurations for the two previously mapped data sets, a Procrustes analysis was performed on the non-metric MDS final configuration with respect to the metric MDS final configuration. The two fitted configurations are then superimposed in a single plot, as shown in figure 4.5. It can be seen that the configurations for both data sets are very similar.



(a) `cmyk` data set.

(b) `animals` data set.

Figure 4.5: Comparison of metric and non-metric MDS final configurations. Crosses represent configuration obtained from metric MDS, circles represent nonmetric MDS configurations. Lines link the two positions of each data point from the two configurations.

This experiment indicates that for data sets in which inter-points distances are all different and so can be ordered in a unique way, metric and non-metric MDS lead to very similar results, but data sets for which many distances are identical, as are data sets with binary features, may have quite different metric and non-metric final configurations. In the general case, it appears that final configurations of metric and nonmetric MDS are often quite identical, so it can be said that metric information can be recovered only from distance orderings. This result was known already in the 60s, and pointed out by e.g. Shepard [118] [119].

4.3 Problems and limitations of the MDS model

4.3.1 Time complexity

The computation time scales as the square of the number of vectors in the data set, and thus places a relatively low maximum limit on the number of objects that may be mapped simultaneously (about one thousand). We can distinguish in the literature two approaches directed at alleviating this problem:

- **subset selection** The idea is to proceed in two steps: First, select from the data set a subset of $N_b\,(N_b \ll N)$ data points (let us call this subset the *base*) and map it using MDS. Second, add somehow the remaining points of the data set to the map of the base subset. This is what Shang and Lee called the *frame method* [23], in which they chose to pick-up randomly the points to form the subset of N_b basis points. The resulting map strongly depend on the chosen heuristic to define the base. A solution that seems more attractive than random selection was presented in *incremental scaling* [6]. The idea is to choose the N_b points with largest edge weights in the Minimal Spanning Tree of the data set. In this way the N_b basis points are the ones that best reflect the global structure of the data.

- **data clustering** Given that in large data sets, many data are similar, it can be decided to first cluster the data and then map (and visualize) only the cluster centers. In his original paper, Sammon suggested to perform a pre-processing of the data using a clustering algorithm such as `ISODATA` when the number of data points exceeds a certain value.

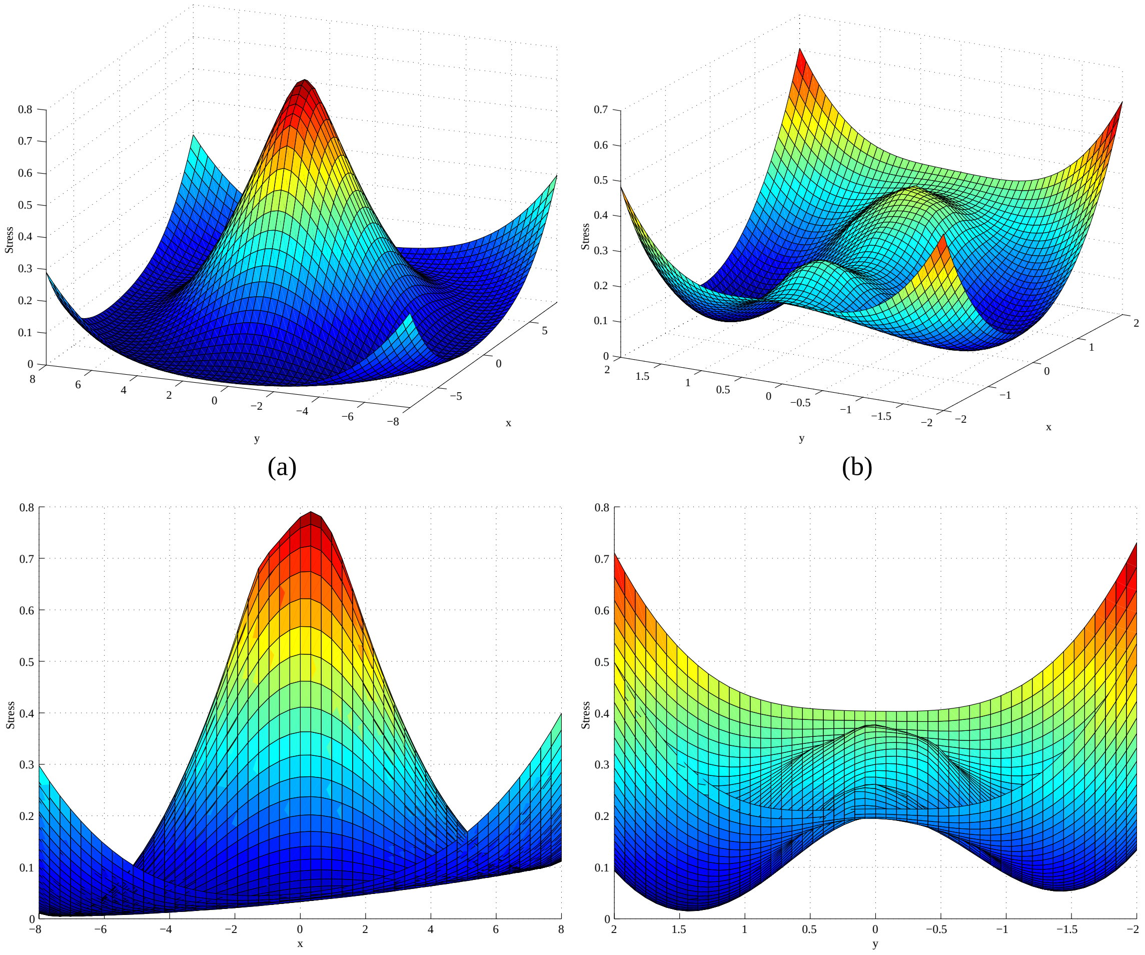
The total computation time also depends on the applied minimization technique. Steepest descent (SD) and the conjugate gradient (CG) techniques are known to be fast. The *coordinate descent* method (CD) introduced by Niemann [102] is even faster and is reported to have excellent convergence properties [44]. The idea is to descend in the direction of the coordinates cyclically (at each iteration another coordinate is chosen), and to choose the step-size by *line search*. In the case of the Stress function, this amounts to minimize a 4-th order polynomial, or a third order polynomial to be zeroed, which can be done analytically. Although this method is fast, it leads to configurations with larger distortions than CG or SD.

4.3.2 Sensitivity to initial configuration

As it was mentioned in §4.2.2, the quality of the resulting configuration of points highly depends on the initial configuration. This feature, shared with the SOM algorithm but not with the PCA approach, is to be related to the iterative nature of those algorithms.

4.3.3 Local minima

The optimization techniques generally used to minimize the loss function are able to find a *local minimum*. Optima with a small *domain of attraction*, if they occur, are likely to be missed. Hence depending on the starting point of the minimization process, the method will end up in a local minimum that can be better or worse. This problem is enhanced by MDS loss function, because of the great number of local minima. In the case of unidimensional scaling (mapping into a one dimensional space), it has been shown [109] that the number of local minima is in general upper bounded by $N!$, and is equal to $N!$ under certain conditions on the given dissimilarities. It seems natural that the number of local minimums will be accordingly larger when mapping in spaces with more than one dimension. In order to have an idea of how such local minimums look like, we plot in 3 dimensions the Stress values as a function of one representative point's coordinates in the map space. In figure 4.6 two different Stress functions are represented by 3-dimensional views of their surfaces in the neighborhood of such local (or global) minimums.



(a)

(b)

(c) `iris`: The ring shaped valley is leaning.

(d) `animals`: Two local minima can be seen.

Figure 4.6: 3-dimensional "views" of Stress functions: Axes *x* and *y* represent one point's coordinates in the 2D space, the other points from the data set are fixed.

4.3.4 Lack of explicit mapping function usable for new data

LSS algorithm does not provide an explicit mapping function governing the relationship between patterns in the input space and in the projected space. Therefore it is impossible to decide where to place new input data in the projected or configuration space. In other words, LSS has no generalization capability. To project new data, one has to run the program again on the pooled data (previously mapped data and new data).

4.4 Proposed improvements

4.4.1 Least Squares of squared distances

In order to alleviate the problem of time complexity of the algorithm, the Stress function can be re-written as follows:

$$SS(\mathbf{Y}) = \frac{1}{F_n} \sum_{i<j}^{N_t} w_{ij} \cdot \left(\delta_{ij}^2 - d_{ij}^2(\mathbf{Y})\right)^2 \tag{4.16}$$

The ALSCAL algorithm is based on such a measure where it is called `SStress`. This expression should lead to final configurations similar to the ones from the normal expression of equation 4.3, probably with better preservation of larger distances due to the squares. The use of squared distances instead of distances make the calculations simpler in the case where the $\{d_{ij}\}$ are Euclidean distances. Expressions for the derivatives with respect to the coordinates vector Y will also be simpler, hence faster calculations of the gradient and optimized step-size. Analytical expressions similar to the ones for the Stress function are presented for the SStress in Appendix A (§A.2). We can see that the number of loops (ie. the summing terms) are identical in both cases for all the expressions. The real gain in calculation time comes from the absence in the SStress expressions of δ_{ij}/d_{ij}^3 and $(\delta_{ij} - d_{ij})/d_{ij}$ terms that are replaced by a constant $(= 2)$ and term $(\delta_{ij}^2 - d_{ij}^2)$, those last terms being already calculated during the SStress function evaluation.

4.4.2 Choice of initial configuration

Empirical experience on various data sets showed that the best Stress reached after a sufficient number of random initializations of the configuration is often lower than the Stress resulting from the mapping initialized by the two first Principal Components, as shown in figure 4.7. When the number of points to be mapped is not too large, it is possible to re-run the MDS algorithm several times and keep the result yielding the lowest Stress. A possible strategy to obtain a reasonably low minimum is to run the minimization algorithm with different random initial configurations as long as the same lowest minimum value is obtained, e.g. for 30 % of the trials. Then we can assume that this minimum is the global minimum. This approach, called sometimes *Multistart* is guaranteed to find a global minimum, albeit after infinitely many random trials, which is its weakness. Furthermore, multistart gives an indication of the *region of attraction* of a (global) minimum, which is the space from which all searches lead to the same minimum. If the data set is too large to allow several mapping trials, it is reasonable to initialize the configuration by the two first Principal Components of the data set. Possible strategies for the initialization are to take the first principal axes directly or after mapping the data in 1-dimensional spaces initialized by the principal axes [50]. In the case of a single point mapped relatively to an existing map, a better choice is to initialize the representative point by the so called *triangulation method* [92], leading to an exact preservation of the distances to the d closest fixed points in the D-dimensional data space $\mathcal{D}$.

4.4.3 Global minimization of Stress

In order to face the problem of local minima, some techniques from *global optimization* can be used because they do not get stuck in local minima. The main drawback of those techniques is that they are generally very time consuming. We have attempted to apply the stochastic Simulated Annealing (SA) method [75] combined with the downhill simplex method of Nelder

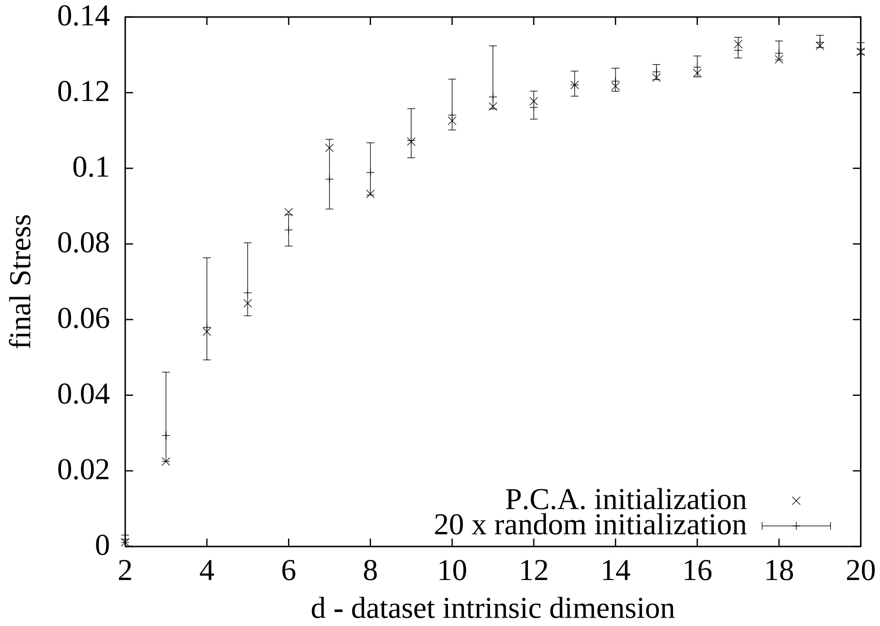


Figure 4.7: Comparison of final Stress values reached after random and Principal Components initializations.

and Mead as proposed in [110, §10.9] to the minimization of the Stress. Experiments were conducted on a few artificial data sets by comparing the final Stress value obtained after SA minimization and optimized step-sized steepest descent. It appeared that SA minimization did never find a lower minimum than the best one obtained from optimized steepest descent minimizations (after several trials with random initializations). The stochastic Simulated Annealing technique has been employed in MDS as well by Klein and Dubes [76], but they conclude that the computational cost of simulated annealing makes it impractical, especially for small problems. Simulated annealing has also been reported to be less efficient than a Molecular Dynamics particles approach in [44]. Global minimization of Stress should be continued in future work because of the increasing computation power of computers that make those techniques more and more accessible.

Various techniques designed at finding the global minimum of the Stress have been presented by different authors, among others: the Iterative Majorization [139] [67] and a statistical mechanics approach [68] [77].

4.4.4 Improvement of convergence by step-size optimization

Kruskal's ad hoc rule (expressed in equation 4.14) for the determination of the step-size $\alpha(t)$ works fairly well for small values of d, but the algorithm becomes very sensitive to α for larger values of d. It has been experimentally found in [30] that for the 3 data sets used in the experiments, when $d > 10$, the traditional algorithm of Kruskal does not always converge. For this reason, and also because Kruskal's rule for the computation of α is not mathematically optimal, we implemented an optimized formula for the calculation of α. The idea [43, pp. 140-141] is to derive an optimal step-size α using an approximation of the minimized function by its second order Taylor expansion.

Optimal step-size $\alpha^{(k)}$ for steepest-descent method

At iteration k, the representative configuration is held in vector $Y^{(k)}$. The steepest-descent method applied to the minimization of a function $S(Y^{(k)})$, which gradient is noted $\nabla S^{(k)} = \nabla S(Y^{(k)})$, is based on the following equation:

$$Y^{(k+1)} = Y^{(k)} - \alpha^{(k)} \cdot \nabla S^{(k)} \tag{4.17}$$

or

$$Y^{(k+1)} - Y^{(k)} = -\alpha^{(k)} \cdot \nabla S^{(k)} \tag{4.18}$$

The second order Taylor expansion of S expressed in vector form is

$$S(Y) \approx S(Y^{(k)}) + {\nabla S^{(k)}}^T \cdot \left(Y - Y^{(k)}\right) + \frac{1}{2}\left(Y - Y^{(k)}\right)^T \cdot H_S^{(k)} \cdot \left(Y - Y^{(k)}\right) \tag{4.19}$$

for Yin the neighborhood of $Y^{(k)}$. At iteration $k+1$, $Y = Y^{(k+1)}$, then (4.19) gives

$$S(Y^{(k+1)}) \approx S(Y^{(k)}) + {\nabla S^{(k)}}^T \cdot \left(Y^{(k+1)} - Y^{(k)}\right) + \frac{1}{2}\left(Y^{(k+1)} - Y^{(k)}\right)^T \cdot H_S^{(k)} \cdot \left(Y^{(k+1)} - Y^{(k)}\right) \tag{4.20}$$

Replacing $Y^{(k+1)} - Y^{(k)}$ expression in (4.18) into (4.20) gives:

$$S(Y^{(k+1)}) \approx S(Y^{(k)}) - \alpha^{(k)} \cdot \left\|\nabla S^{(k)}\right\|^2 + \frac{1}{2}\left(\alpha^{(k)}\right)^2 \cdot {\nabla S^{(k)}}^T \cdot H_S^{(k)} \cdot \nabla S^{(k)} \tag{4.21}$$

$S(Y^{(k+1)})$ is minimized with respect to $\alpha^{(k)}$ by the choice

$$\frac{\partial S(Y^{(k+1)})}{\partial \alpha^{(k)}} = 0, \tag{4.22}$$

Applying (4.22) to (4.21) gives

$$\frac{\partial S(Y^{(k+1)})}{\partial \alpha^{(k)}} \approx -\left\|\nabla S^{(k)}\right\|^2 + \alpha^{(k)} \times {\nabla S^{(k)}}^T \cdot H_S^{(k)} \cdot \nabla S^{(k)}, \tag{4.23}$$

Finally (4.22) and (4.23) lead to the result

$$\alpha^{(k)} = \frac{\|\nabla S^{(k)}\|^2}{{\nabla S^{(k)}}^T \cdot H_S^{(k)} \cdot \nabla S^{(k)}} \tag{4.24}$$

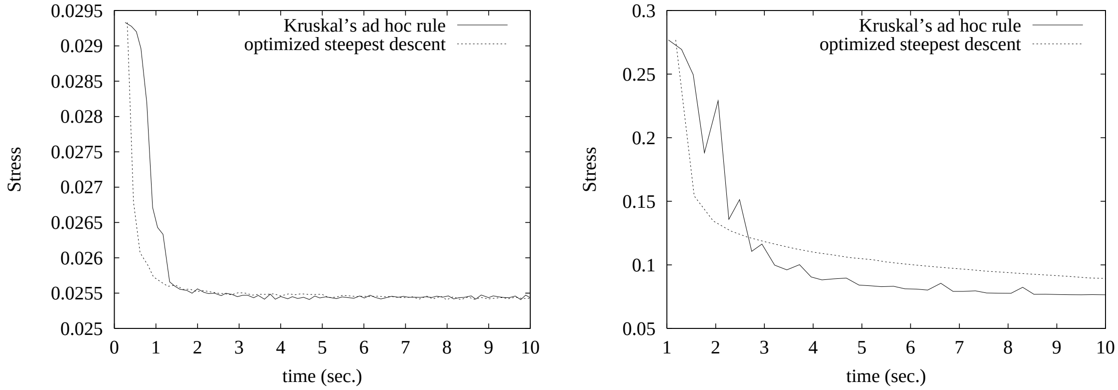
When ignoring steepest-descent equation (4.17) and choosing $Y^{(k+1)}$ to minimize directly the second order expansion of S expressed in (4.19), we obtain the expression used in Newton's method:

$$Y^{(k+1)} = Y^{(k)} - {H^{(k)}}^{-1} \cdot \nabla S^{(k)} \tag{4.25}$$

As we shall see from the detailed analytical calculations presented in Appendix A, the computation of the optimal step-size $\alpha^{(k)}$ of expression (4.24) does not require an explicit calculation of the Hessian matrix $H_S^{(k)}$ because of simplifications occurring in the product ${\nabla S^{(k)}}^T \cdot H_S^{(k)} \cdot \nabla S^{(k)}$. After analytical simplifications of $\alpha^{(k)}$'s expression, its computation scales with d^2N^2 [10]. Many

[10] Computation complexity of α_n is $\mathcal{O}(N_m \cdot d_{out})$ and of α_d is $\mathcal{O}(N_m \cdot (N_m + N_f) \cdot {d_{out}}^2)$.

calculations needed to the evaluation of optimized step-size are shared with calculations needed to the evaluation of the Stress function $S^{(k)}$ or its gradient $\nabla S^{(k)}$. Therefore the computer cost of step-size evaluation is not much higher when using optimized α than when using Kruskal's α. As shown in figure (4.8), the great advantage of optimized step-size minimization over Kruskal's is that Stress decrease is much smoother, eliminating jumps upwards of Stress. Such "peaks" appear especially when the configuration is near a local minimum, which is often the case after configuration initialization by PCA. Those peaks, although they are sometime helpful because they allow to "jump" to a lower basin of attraction, make difficult the stopping of the iterations. During a complete minimization process, the time saved by the fact that α is more optimal and the minimization more efficient at each step is greater than the time spent in all evaluations of step-sizes, so that the minimization process is at the same time shortened and its convergence is improved. The plot of the evolution of Stress during minimization shown in figure (4.8) confirms that convergence of the minimization process is improved by step-size optimization while performance is *most of the time* as good as with Kruskal's step-size.



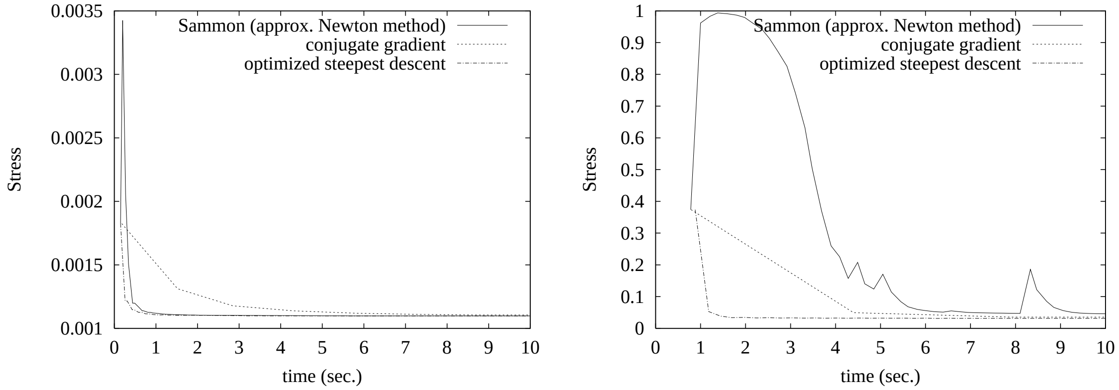
(a) `iris` data set, initializations by PCA.

(b) `brod2` data set, initializations by PCA.

Figure 4.8: Comparison of Stress minimization by Kruskal's or optimized step-size.

A performance comparison of the optimal step-size steepest descent method with other optimization techniques

Minimizing a multivariate function such as the Stress function can be achieved using various methods from the unconstrained optimization family of techniques. A number of methods belonging to 4 classes is usually referred to [110], differing in the way they use the first and second order derivatives of the minimized function: steepest descent methods, conjugate gradient methods, Newton's method and quasi-Newton methods. Experiments were conducted in order to check how the steepest descent with optimized step-size for Stress minimization compares to a conjugate gradient method and to the original Sammon's method (ie. an approximated Newton method). It appeared that our method is the fastest of the three methods, and more efficient (it gives comparable or slightly lower minima). Stress minimization performances for two data sets are given in figure (4.9). The first one is `iris` and the second one is `brod2` [30], a data set (obtained through a wavelets decomposition of a Brodatz album image, see [30]) containing 245 vectors described by 39 features, the intrinsic dimension was estimated to be $D_i \simeq 3$. It can be seen that the minima reached by all three methods are comparable, but the computation time needed to reach this minimum is shorter for the optimized steepest descent than for conjugate gradient or Sammon's method. Jumps upwards of the Stress value during the first steps of the iterative minimization may occur also with optimized step-size, although much more rarely than with constant step-size along the gradient. These jumps just mean that the current point



(a) `iris` data set, initializations by PCA.

(b) `brod2` data set, initializations by PCA.

Figure 4.9: Comparison of Stress minimization by approximate Newton method (Sammon's), conjugate gradient and optimized steepest descent.

$X^{(k)}$ is too far from a local minimum, so that the second order Taylor expansion of the Stress function $S(X^{(k)})$ is not a good approximation of this function in this neighborhood.

In order to summarize the reasons why to use MDS with optimal step-sized steepest descent instead of a pure Newton's method, we can say that Newton's method:

- gives a greater improvement per step than the steepest descent
- needs intensive computation of H^{-1}
- is well suited to the high-accuracy location of an optimum near which quadratic approximation is good

On the other hand, steepest descent with optimal step-size is a compromise that:

- takes into account the curvature of $S^{(k)}$ at $Y^{(k)}$ (second order derivatives)
- is not as computationally intensive as a real second order method
- is more efficient than a conjugate gradient method.

4.4.5 Mapping new data using "relative" mapping

Suppose we are now in the following situation: We know the mapping of a given data set (this map may be obtained by MDS or any other technique), after which we receive new data (one point or more) and we want to see where they would be placed with respect to the mapped points from the data set. Let us first define the following notations: $\{X^n\}$ is the set of N_n new points to be mapped and $\{Y^n\}$ their positions on the map, $\{X^b\}$ is the set of N_b data points already mapped, called the *base*, and $\{Y^b\}$ their positions on the map. We remarked in §4.3.4 that MDS does not provide any explicit mapping function that would project new data on an existing map, and that the only solution is to pool the new data $\{X^n\}$ with the one already mapped $\{X^b\}$ in one common set and run the algorithm on this pooled set. This approach suffer the inconvenience to repeat the calculations needed to find the positions of the points $\{Y^b\}$ that were already mapped. Another disadvantage is that the new points $\{Y^n\}$ will interfere with points $\{Y^b\}$ during the mapping process to find their optimal position on the map. The configuration $\{Y^b\}_2$ obtained by mapping points $\{X^b\}$ together with the new points $\{X^n\}$ will then differ from the configuration $\{Y^b\}_1$ that we had before (this difference is increasing with the ratio N_n/N_b).

The following simple modification to the Stress function will let the configuration $\{Y^b\}_2$ be identical to $\{Y^b\}_1$: in the standard Stress expression (4.3), remove the distances between the points $\{Y^b\}$ so that these points will be kept fixed during Stress minimization. The pooled data set is initialized as follows: points $\{X^b\}$ are initialized by $\{Y_b\}_1$ and the new points $\{X^n\}$ are initialized in one of the following ways[11]:

- each point P_i^n is initialized individually by linear interpolation between its 2 nearest neighbors denoted (P_1^b, P_2^b) among the points $\{P^b\}$. Denoting $d_{n1} = d(P_i^n, P_1^b)$ and $d_{12} = d(P_1^b, P_2^b)$, $\{Y_i^n\}$ is found from the relation $\overrightarrow{P_1^b P_i^n} = \frac{d_{n1}}{d_{12}} \overrightarrow{P_1^b P_2^b}$.
- each point P_i^n is initialized randomly.

In other words, as the primary purpose of the Stress function is to perform a least square minimization on the distances, it only needs to sum over the distances that are changing during minimization. As some distances are constant during Stress minimization (the distances between fixed points $\{Y^b\}$), the Stress expression can be redefined as

$$S_r(\mathbf{Y}) = \sum_{i<j}^{N_m} w_{ij} \cdot \left(\delta_{ij} - d_{ij}(\mathbf{Y})\right)^2 + \sum_{i=1}^{N_m} \sum_{j=N_m+1}^{N_t} w_{ij} \cdot \left(\delta_{ij} - d_{ij}(\mathbf{Y})\right)^2 \tag{4.26}$$

in which the pairs of points taken into account in the sum are all the pairs of new points $\{X_i^n, X_j^n, i \neq j\}$ plus all the pairs $\{X_i^n, X_j^b\}$. The ensuing mapping process is called *relative MDS mapping* because the new points $\{X^n\}$ are mapped relative to the points $\{X^b\}$.[12] A scheme of relative MDS mapping as compared to standard MDS mapping is given in figure 4.10. The algorithmic complexity of relative mapping can be reduced by taking into account even less distances in the Stress expression, for instance by taking, for each new point, its distances to its k nearest neighbors in the basis of fixed points (instead of all the points of the basis). As will be shown in chapter 6 on applications of MDS, relative mapping can be used to visualize large data sets in the following manner: first clusterize in some way the data points to reduce their number in order to allow the mapping of the clusterized points using standard MDS, then add to the clusters map the original points using relative MDS.

Other approaches to add generalization capability to MDS

A first simple idea to provide MDS with generalization capability is the following: First map the data using any MDS algorithm, and then build a classical neural network (e.g. MLP with linear transfer functions in output layer units) that learns the produced 2-dimensional points coordinates on the basis of the D-dimensional input data.

A method called *distance mapping* was proposed in [108]. We denote D_b the distance matrix for the points of the base in the input data space, and as before Y_b is the coordinates matrix of the base. First we define matrix V such that:

$$D_b \dot{V} = Y_b \tag{4.27}$$

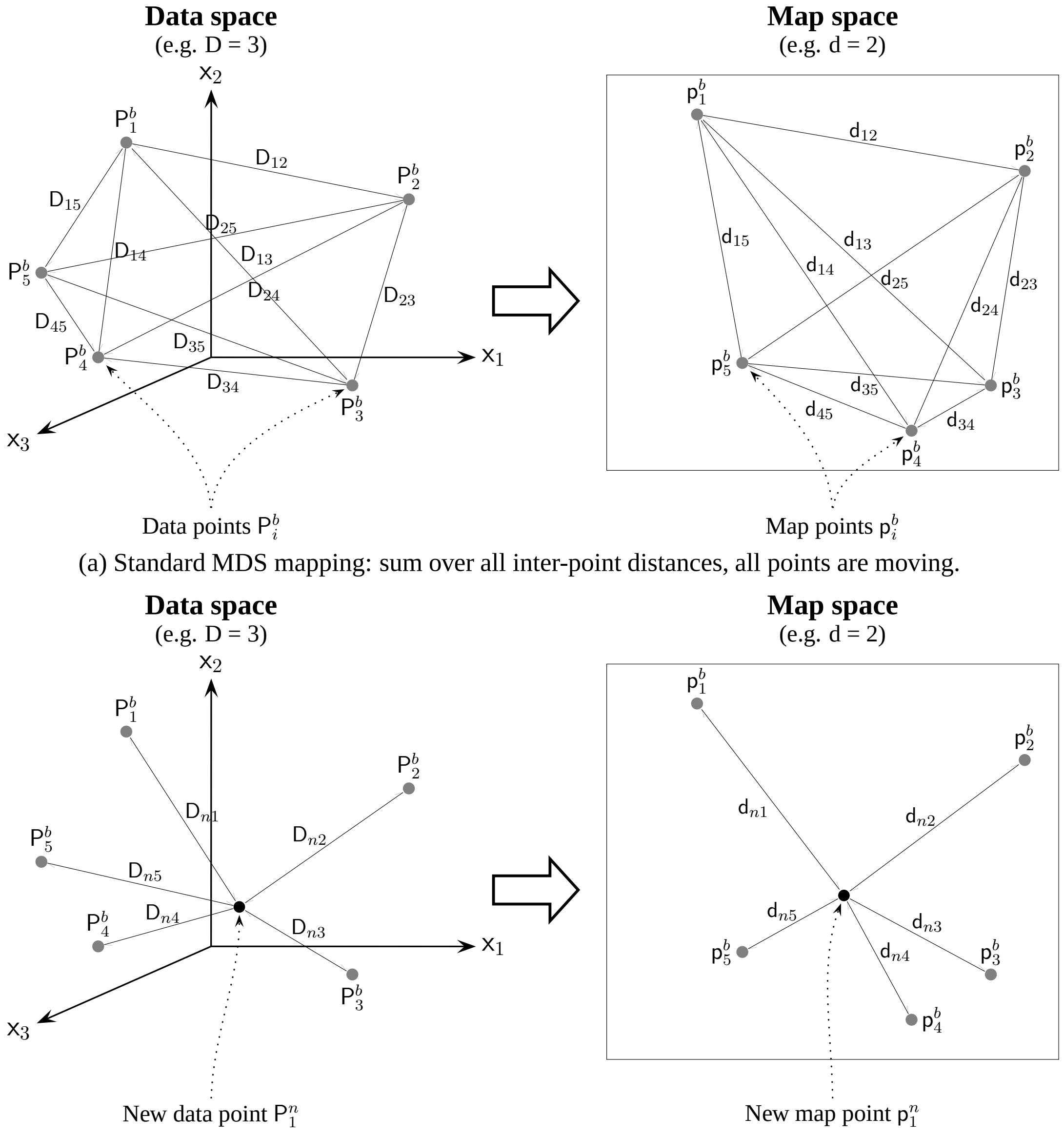
Matrix V necessarily exists because matrix D_b is full rank. Then we derive matrix Y_{new} of the new data points coordinates using:

$$Y_{new} = D_{newtobase} V \tag{4.28}$$

where $D_{newtobase}$ is the matrix of the distances between points in X_{base} from points in X_{new}.

[11] Initialization by a PCA of the pooled data set would require a kind of Procrustes analysis to make the initial positions of the fixed points derived from the 2 first PC match as well as possible to the $\{Y_b\}_1$. This solution needs many calculations.

[12] The idea of minimizing the Stress function was presented by Demartines in his PhD thesis [34] (where it is called *interpolation*) in order to provide generalization capability (continuity) to his mapping.



(a) Standard MDS mapping: sum over all inter-point distances, all points are moving.

(b) Relative MDS mapping: sum over all distances to new point, only point p_1^n is moving.

Figure 4.10: Mapping new data using “relative” mapping.

4.4.6 Zooming on subspaces using “localized” mapping

In the case of database exploration, it is desirable to focus on chosen areas of the data space. Such an area can be defined interactively by choosing one data point P_c and zooming its data neighborhood $\{P_{N_c}\}$, that is the part of the data space that contains the k–nearest neighbors of P_c, or the data points within a neighborhood of given radius r. As our interest is the neighborhood of point P_c, the distances from P_c to the points from $\{P_{N_c}\}$, as well as distances separating points close to P_c, should be primarily preserved. This special feature of the mapping can be realized by choosing properly the weights in the Stress expression. Our proposal is to choose a Gaussian-like term centered on point P_c as a functional of the decreasing mean distance $D_{cij} = (D_{ci} + D_{cj})/2$ between D_{ij} end points (P_i and P_j) and point P_c, expressed as:

$$w_{ij} = \frac{1}{\sigma\sqrt{2\pi}} \cdot e^{-\frac{D_{cij}^2}{2\sigma^2}}, \tag{4.29}$$

Parameter σ allows the controlling of the width of the neighborhood with better topology preservation and it must be adjusted manually when zooming. As shown in [101], this technique can improve the resulting mapping, although not necessarily, depending on the value choosen for σ.

Chapter 5

A comparison of MDS and SOM algorithms in practice

Data topology preservation

The two approaches presented above are non-linear mappings, they are hence able to display non linear data manifolds, which cannot be realized using the linear PCA. For a given data set, there is no one projection that is better than all the others. Each approach presented in this thesis is optimal in its own particular way. It is not possible to point out universal mapping technique, since the mapping effectiveness varies with the data structure and the user's expectations. The following quotation of Siedlecki et al. [120] affirms this point of view :

> "In attempting to evaluate mapping techniques, we have to agree that the dimensionality reduction techniques, if used to create two-dimensional displays, serve a human observer; therefore, the human observer is uniquely qualified to decide whether the display is good or useless for a particular application. [...] Since human intuition plays the main role in this data analysis process, it seems unreasonable to evaluate display produced by mapping methods by directly applying numerical criteria to the displays, for these criteria cannot fully reflect the overall information carried by these displays."

It follows that comparing mappings obtained by SOM and MDS using e.g. their Stress values would lead to a false conclusion that MDS is better than SOM because the Stress is evidently better. Siedlecki et al. compared in their study a set of mapping techniques on the basis of several people analyzing independently the mapping results produced for several different data sets.

We conducted various experiments in order to show how MDS and SOM compare from the point of view of data topology preservation when visualizing multivariate data. The well known `iris` data set was previously displayed using SOM in figure 3.3 (page 18), and using MDS in figure 4.1 (page 33). It can be noted that SOM fails to capture the global relationships between the three clusters, and the local dispersion of the clusters. Experiments designed to see how the algorithms manage to map simplexes [42] showed that the symmetry existing in N-dimensional simplexes is much better rendered by MDS than by SOM, as shown in figure 5.1(b). Similar experiments on mappings of spheres or hyper cubes (see figure 5.1(a)) [40] lead to the same conclusion as reported in [41].

Another example showing the differences between mappings obtained from the two methods is the `color` data set, consisting of 124 colors described by 3 numbers (floating points scaled between 0.0 and 1.0, instead of integers between 0 and 255), corresponding to the (R,G,B) levels of the colors. The reason for using color data is that human can easily check perceptually inter-colors similarities if each color data is marked using the color it represents. The mappings presented on figure 5.1(c) show that both methods exhibit topology preservation (the colors are varying continuously on both maps), but the SOM mapping is “stretched” to fit the given array of nodes.[1]

Feature extraction capability

As mentioned in §1.3, a reduction of the dimensionality of multivariate data can enable the application of pattern recognition procedures and avoid the “curse of dimensionality” problem. It is interesting to check how much DR affects the classification performances of a given classifier. The effect of the reduction of data dimension by one linear (PCA) and four non-linear techniques (SOM, MDS, NLM, AFN) was compared from the viewpoint of classification performance using a k–NN classifier ($k = 5$) with the leave-one-out technique, on one artificial and two gray-level and color texture data sets [30] [117]. This study showed that all four non-linear DR techniques lead to better classification results than PCA, and that data reduced by neural networks (SOM, AFN) give better classification results than those obtained by statistical mapping methods (MDS, NLM), especially when the number of reduced dimensions is small ($d < 4$).

Generalization capability

An important feature to the credit of neural networks in general is that they offer an explicit mapping of data points, allowing thereby the mapping of new points. For instance points from a test set unused during the training stage of the network can be easily added onto the existing map of the training data. Mapping new points using standard MDS is not possible directly. The additional feature of mapping with respect to already mapped points is a contribution of this work and is called *relative mapping*.

Memory requirements

The main memory requirement of SOM algorithm is for the codebook storage, whereas MDS algorithm needs at least the dissimilarities and inter-point distances storage. The memory needed for the codebook vectors scales with the number of neurons on the output layer of the SOM and with the data space dimensionality $X_{dim} \cdot Y_{dim} \cdot D$. SOM algorithm will have increasing memory requirements for large maps, which is recommended when the data set size is large or in order to increase data topology preservation. The memory needed for the dissimilarities and inter-point distances storage in MDS scales with their number $N(N-1)/2$. MDS algorithm will have exponentially increasing memory requirements when applied to large data sets. For very large databases, SOM is better suited to the visualization because (due to the fixed number of neurons on the map) it performs a kind of clustering of the data. MDS algorithm being sensitive to the number of points (in terms of memory and time complexity), a separate clustering pre-processing (using for instance the technique applied to the `thyroid` data set in §6.2.1) of the data should be considered.

Time complexity

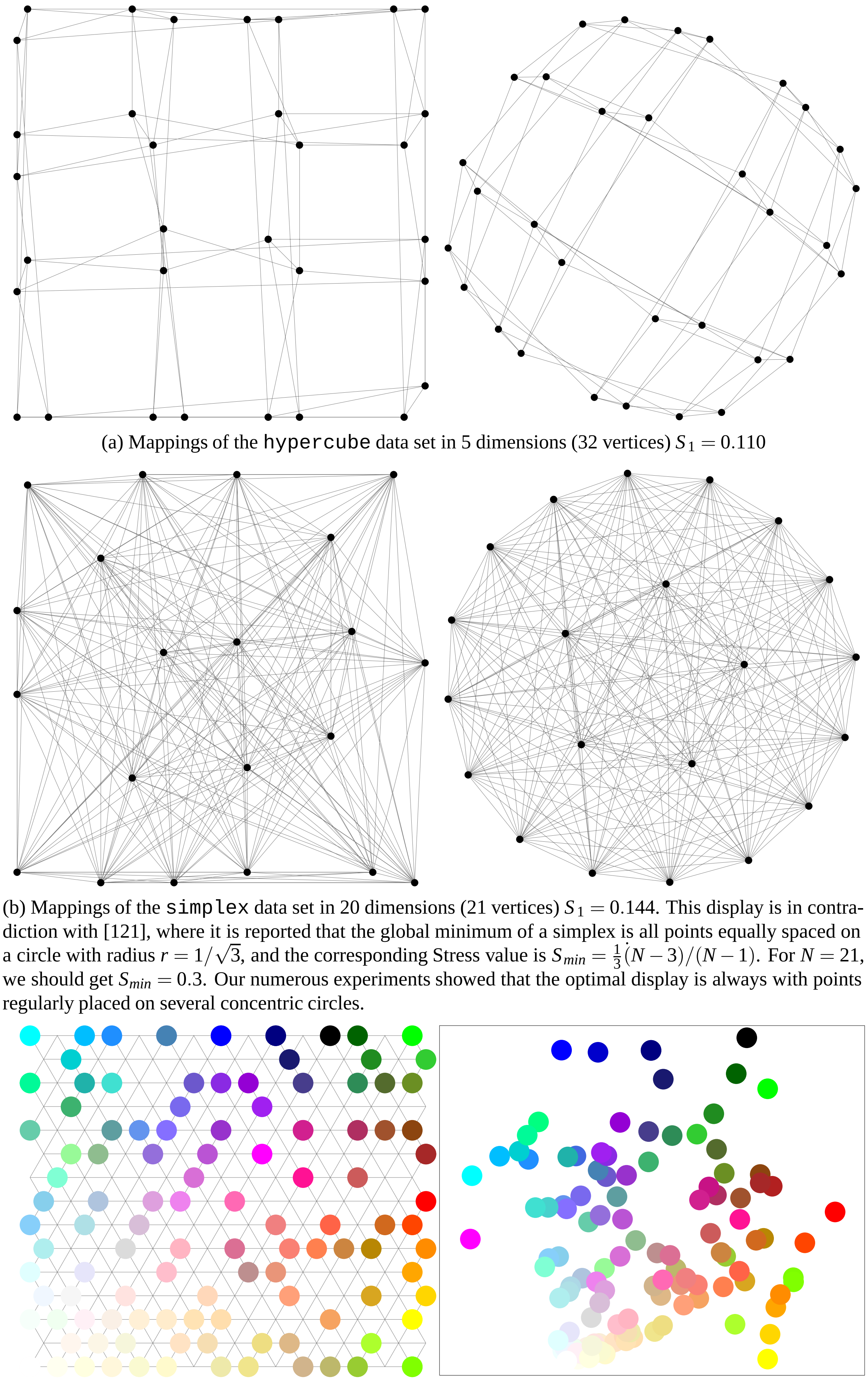
The time complexity of both algorithms is mainly related to the same parameters as for memory requirements. Hence the main computational advantage of SOM over MDS is related to the

[1] Several SOM maps showing colors on the basis of their RGB values are presented in [69], in the purpose of introducing the words category maps on which the two level WEBSOM is based.

number of points that can be mapped, which can be for SOM much larger than for an MDS mapping. On the other hand, the dimensionality of the data has practically no effect on MDS time complexity, but increases the one of SOM.

Typical applications

SOM algorithm can be used in applications which purpose is the visualization of a very large number of data (e.g. databases of texts) or in applications in which new data are created during times (e.g. monitoring of industrial processes [1]). SOM is suited for these data analysis and engineering tasks because they require at the same time a visualization of many database objects and a classification of new incoming data of similar nature. But it must be kept in mind that the performances of SOM for each of those two tasks taken separately are poor. The main advantage of SOM is hence to regroup those two features in one compact tool that is quite small and simple to implement. MDS algorithm was designed as an exploratory data analysis tool which principal feature is data topology preservation that is assorted with a mean to control the quality of this data topology preservation by the design of the Stress function. MDS should be used then in applications processing multivariate data sets of moderate size for which a display showing precisely inter-data relationships is needed. Medical data often possess those features hence the idea to apply MDS in medical diagnosis aid tools.



(a) Mappings of the `hypercube` data set in 5 dimensions (32 vertices) $S_1 = 0.110$

(b) Mappings of the `simplex` data set in 20 dimensions (21 vertices) $S_1 = 0.144$. This display is in contradiction with [121], where it is reported that the global minimum of a simplex is all points equally spaced on a circle with radius $r = 1/\sqrt{3}$, and the corresponding Stress value is $S_{min} = \frac{1}{3}\dot{(}N-3)/(N-1)$. For $N = 21$, we should get $S_{min} = 0.3$. Our numerous experiments showed that the optimal display is always with points regularly placed on several concentric circles.

(c) Mappings of the `color` data set in RGB space (124 colors)

Figure 5.1: A comparison of SOM (left) and MDS (right) mappings for three data sets.

Chapter 6

Applications of MDS

In order to enable the user to see more precisely the global or local data structures of a given data set, an interactive MDS based mapping software has been developed. As we will see in this section, this *interactive exploratory data analysis* tool can be used to first visualize the global structure of a data set, and then *zoom* in chosen areas of the data space.

Data pre-processing

Very often in practical applications, data are pre-processed in order to prepare them to the main procedures. It can be asked which pre-processing is necessary prior to visualization by MDS? For the multivariate data we deal with, such a pre-processing can be just a centering of the data, a standardization, normalization or whitening, or treating in some way the possible missing values. MDS is based on distances, so any transformation that affects distances will modify the visualization. Centering the data is just a shifting of the configuration and has no inference on the distances, so it can be applied prior to MDS. Any pre-processing that changes the relative scales of the features will have a drastic effect on the result if distances are not preserved. It is obvious that standardization does not preserve Euclidean distances, therefore this transformation is not orthonormal [52] and it is not desirable in the case of MDS. In many real-life data such as medical data, the features are made up of various measurements of very different nature (age, temperature, blood pressure, ...) that have different ranges and means. Distances based on such raw data would over-emphasize the importance of features with very large values. This is why standardization is often performed on medical data sets. Data were also standardized in the case of classifier's decision borders visualization, when the classifier itself was trained on standardized data and then new data classified after standardization.

Concerning the missing data occurring in real life data sets, we implemented in our software the following possible treatments. Discard data points that have at least one missing feature, discard the features with at least one missing value, set the missing feature values to the mean feature value averaged on all the data points or within classes. Another strategy that was claimed to perform well in pattern recognition tasks [38] is the following: When computing the distance between two objects, set the missing difference(s) to the mean value of the computed differences for this pair of objects.

6.1 Visualization of psychometric data

6.1.1 Database visualization

An interesting application of our interactive MDS software to the visualization of medical databases or data sets is the visualization of a given new case or patient in relation to known cases from the database.

The psychometric test *Minnesota Multiphasic Personality Inventory (MMPI)* [20] [21] was designed to help practitioners to diagnose psychological diseases. This test consists in asking the patient more than 550 questions about various aspects of his life to which the possible answers are "`yes`", "`no`", "`I don't know`". On the basis of the answers, a set of 13 numerical scales (integers from 20 up to 120) is computed leading to a "profile" of the patient. Then practitioners diagnose the patients, and a set of psychometric nosological types (or classes) can be built. In the case of the data analyzed here below, the classes defined by J. Gomuła and T. Kucharski are the following:

Women types: norm, neurosis, psychopathy, organic, criminality, schizophrenia, reactive psychosis, involutive psychosis, simulation, deviational answering style 1 to 6.

Men types: neurosis, psychopathy, alcoholism, criminality, schizophrenia, reactive psychosis, simulation, deviational answering style 1 to 6.

Common types: norm, psychopathy, drug induction, organic, syndrome delusion, reactive psychosis, paranoia, manic state, simulation, and dissimulation.

The patients are grouped into two databases, according to the patient's gender. The `women` database contains 1711 patients distributed into 20 classes of women and common types. The `men` database contains 1787 patients distributed into 20 classes of men and common types. There is no missing data in these data sets.

The size of each database is small enough to allow its complete mapping in one MDS run, even with the Multistart initialization approach to get the best possible final display[1]. Then if a practitioner gets the psychometric measurements of a new patient, the new data point can be mapped onto the database display using relative mapping, which can be done quickly because just one point is moving during the mapping. In this way, the practitioner obtains immediately an image of the location of his patient in relation to the database known cases, and is able to make a judgement on how to classify this patient. In this manner, the practitioner can use the software as a diagnosis aid tool. Displays of the two databases for `women` and `men` are shown in the following 8 figures 6.1 to 6.8: The first 4 figures represent the `women` database visualized using PCA mapping (fig. 6.1), MDS mapping (fig. 6.2), SOM mapping (fig. 6.3) and the decision borders of the SOM mapping as shown in §3.3.4, fig. 3.6(b) (fig. 6.4). In figures 6.2 and 6.3, a new data (black dot marked by an arrow) has been added to the database maps using relative mapping for MDS and usual data mapping for SOM. On both displays, the black dot is mapped close to data from class `organika` (light blue dots), suggesting such a classification. The last four figures represent the mappings using the same methods for the `men` database. The displays of the databases obtained by the Self-Organizing Maps algorithm are shown here in a purpose of comparison with the ones obtained by MDS mapping. It can be seen that some classes (e.g. `norma`, `psychopatia` and `organika`) are more clearly separated on the MDS mappings (figures 6.2 and 6.6) than on the PCA mapping (figures 6.1 and 6.5).

[1] This was not very useful however because the result obtained after PCA initialization was much better than all the 100 random trials. This may suggest that the iterations were stopped too early, so that the stopping criterion should be made dependent in some way on the data set size.

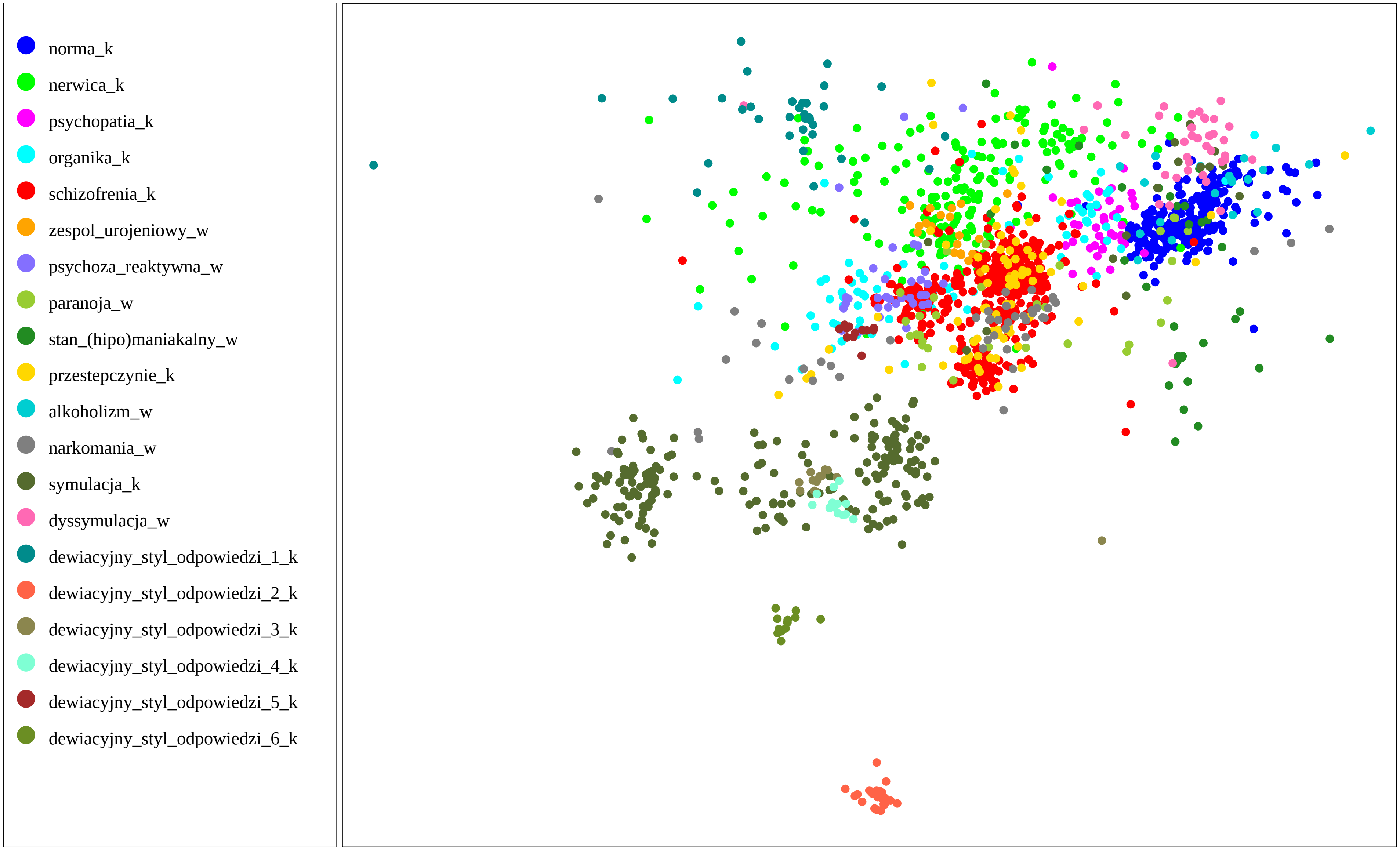


Figure 6.1: Psychometric `women` database visualized using PCA mapping: data points mapped on the two first principal components.

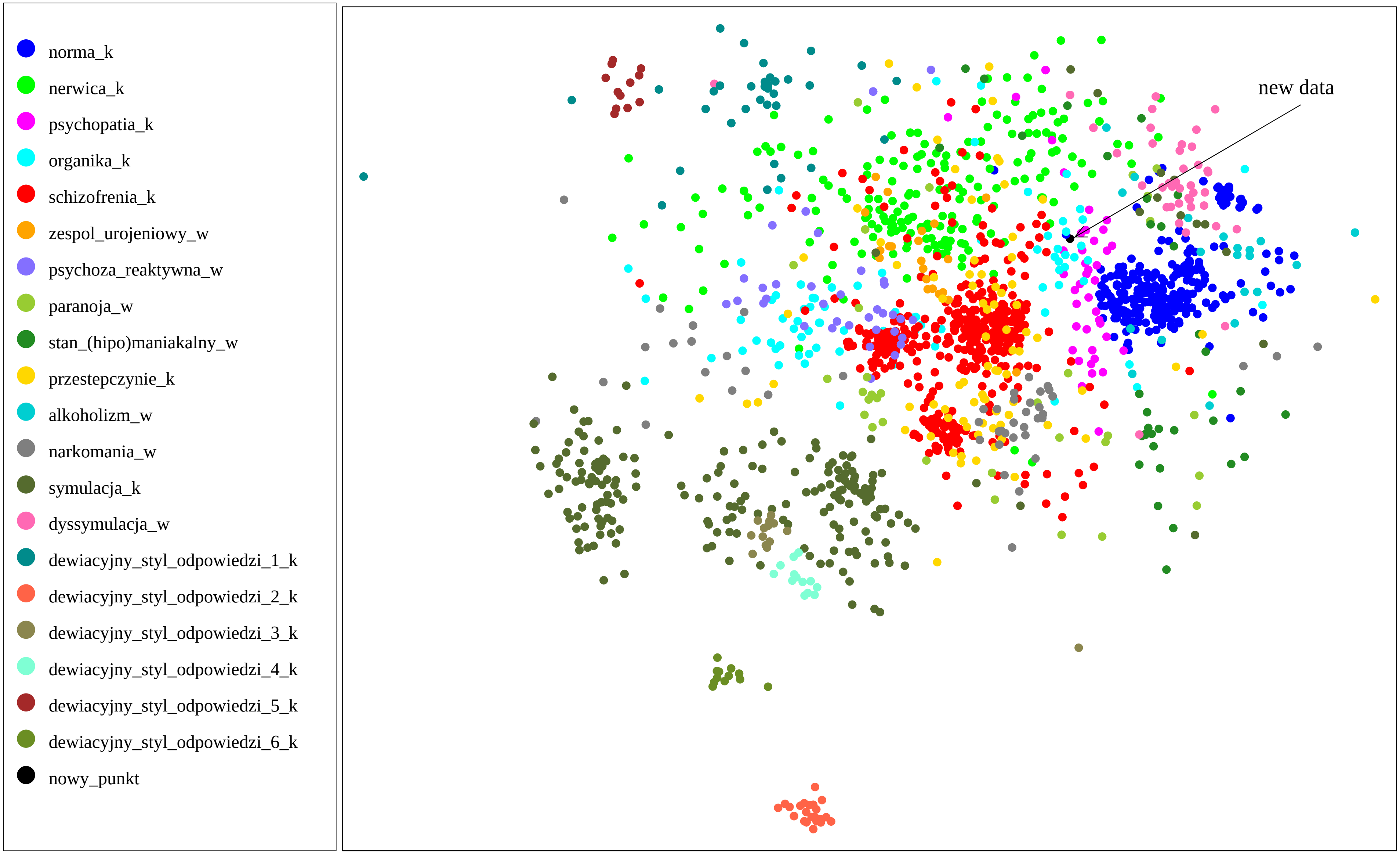


Figure 6.2: Psychometric `women` database visualized using MDS mapping: PCA initialization, final Stress: $S_1 = 0.021$ (142 iterations).

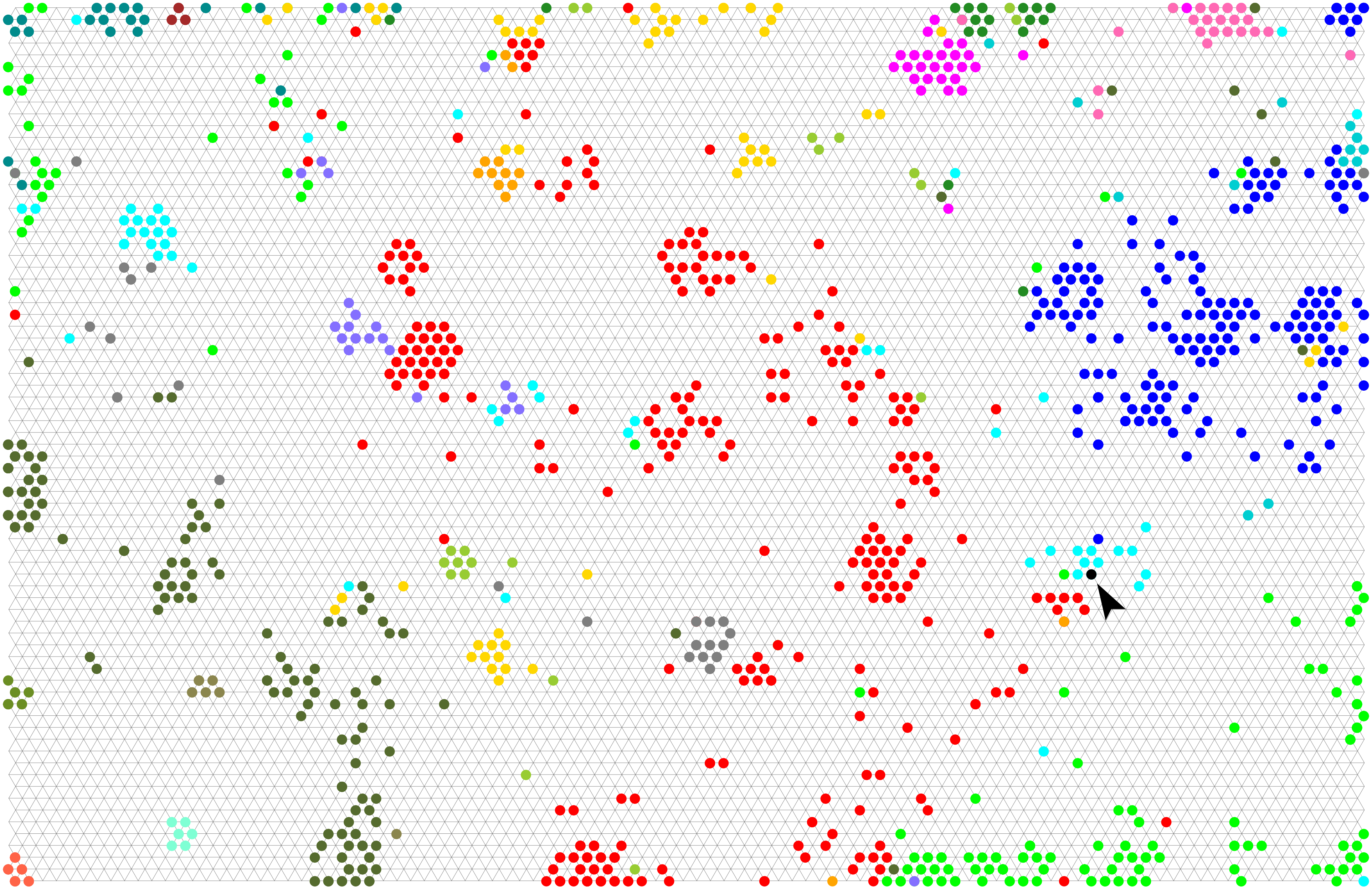


Figure 6.3: Psychometric `women` database visualized using the Self-Organizing Maps mapping: 100 × 75 neurons, random init. nb. 8.

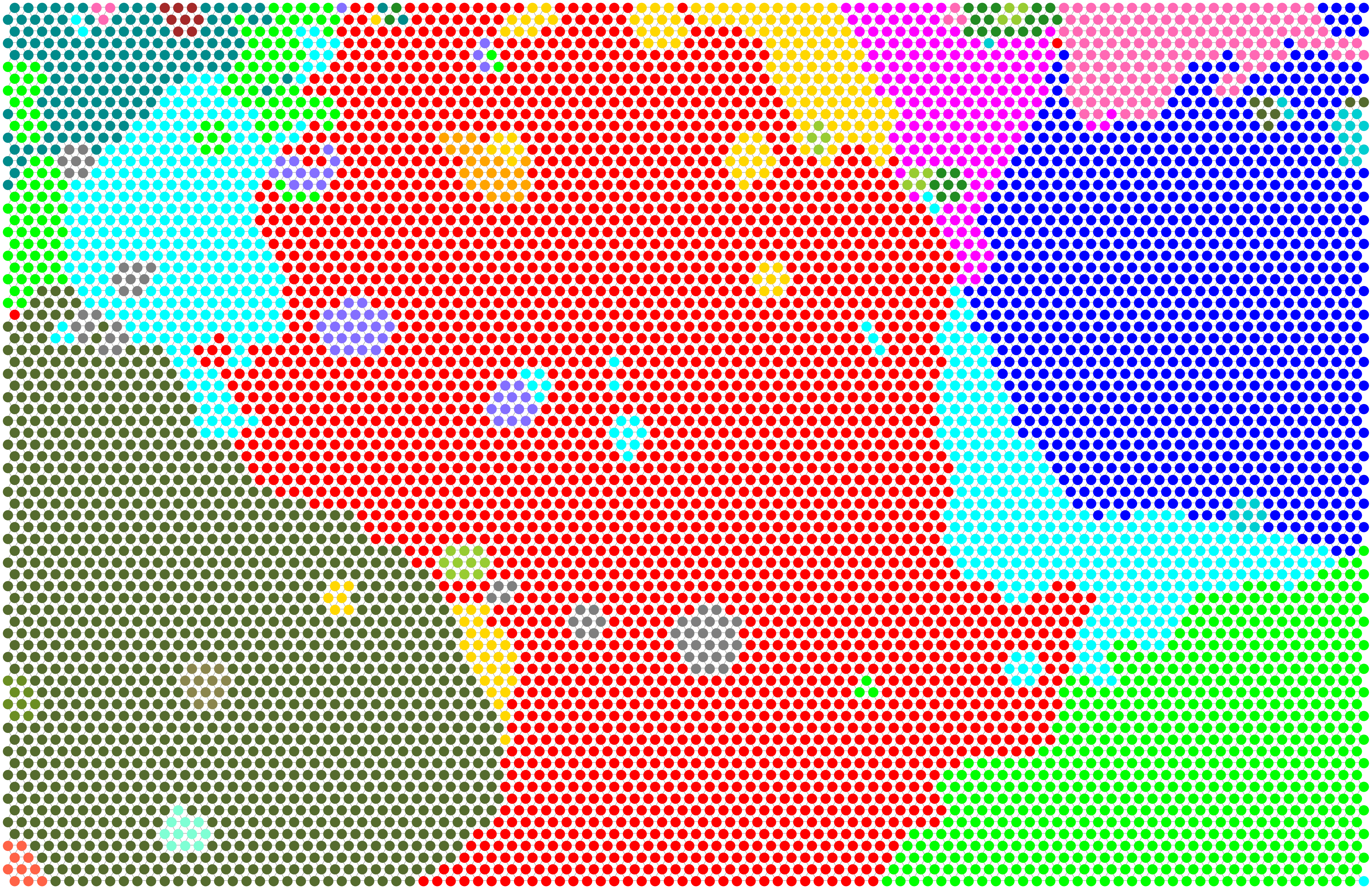


Figure 6.4: Psychometric `women` database visualized using the Self-Organizing Maps mapping: decision borders are visualized.

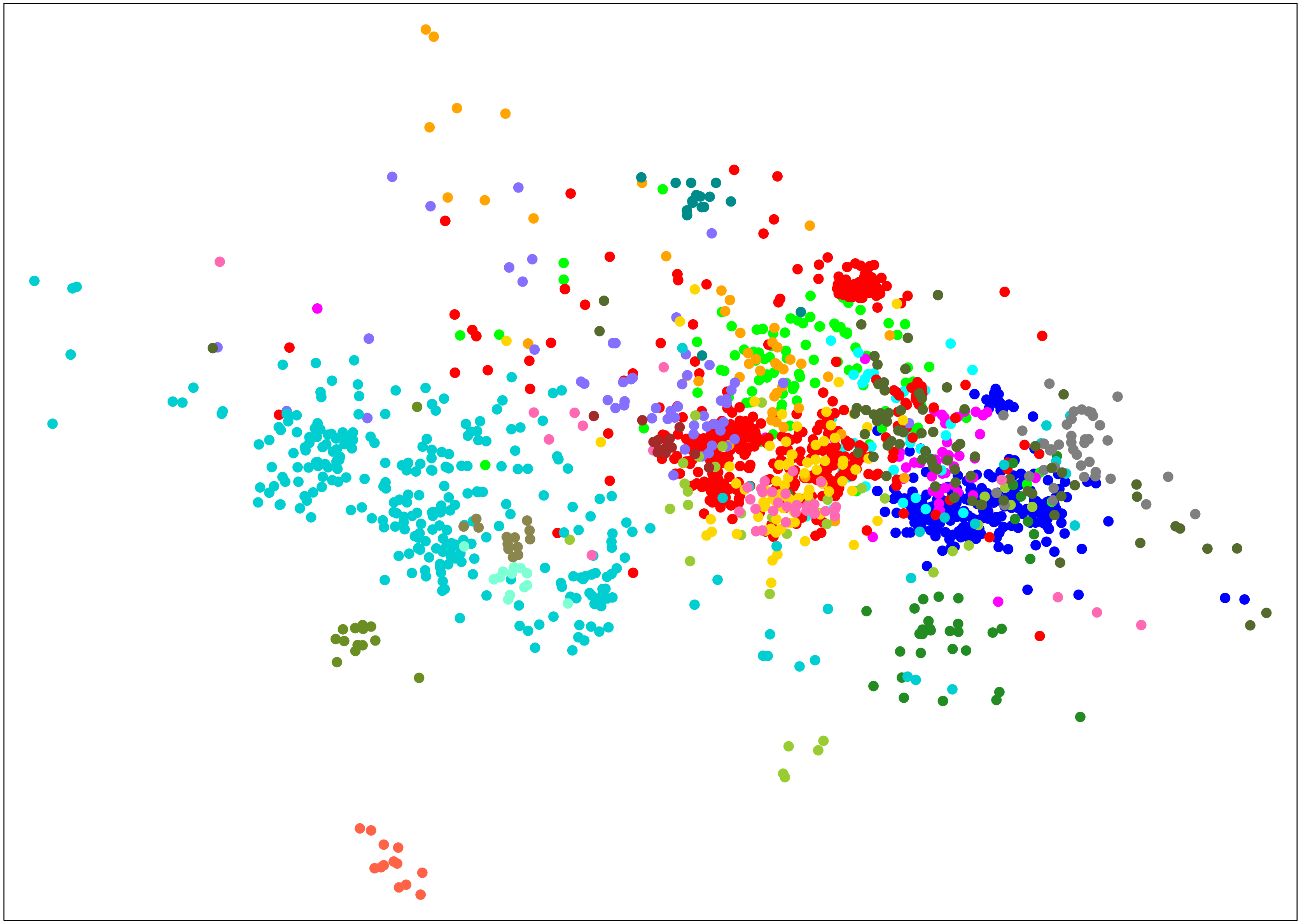


Figure 6.5: Psychometric `men` database visualized using PCA mapping: data points mapped on the two first principal components.

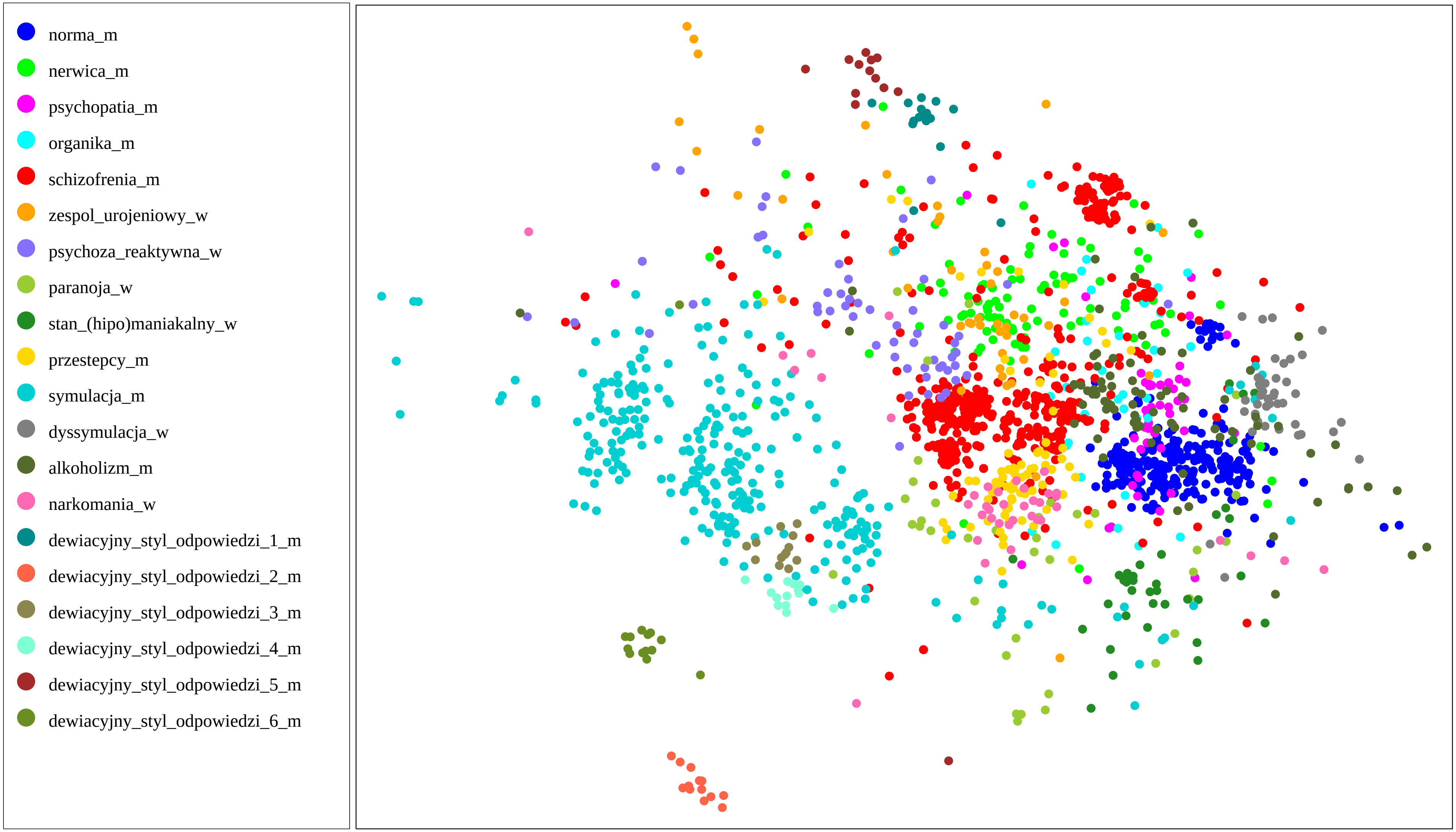


Figure 6.6: Psychometric `men` database visualized using MDS mapping: PCA initialization, final Stress: $S_1 = 0.018$ (471 iterations).

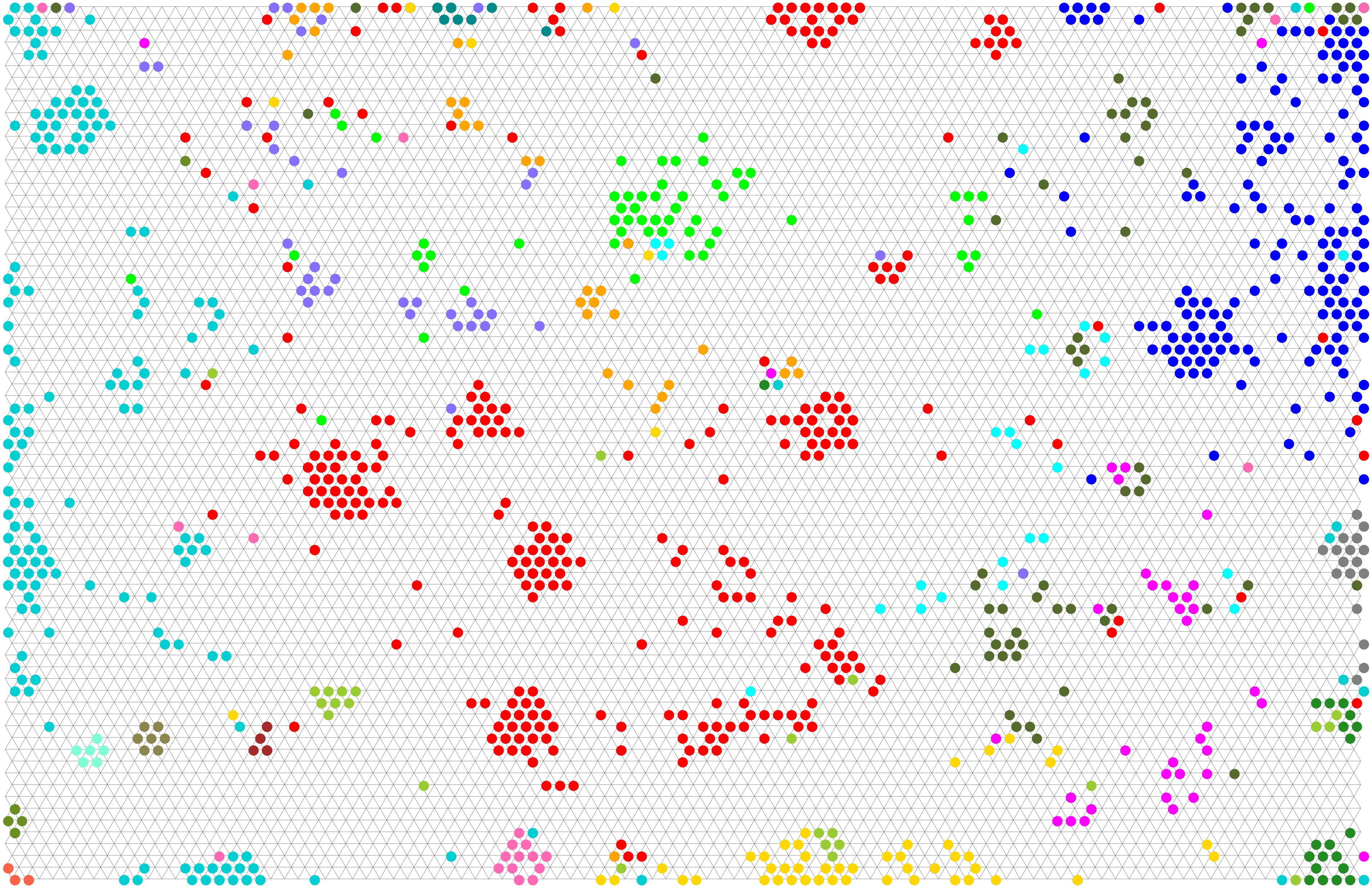


Figure 6.7: Psychometric `men` database visualized using the Self-Organizing Maps mapping: 100 × 75 neurons, PCA initialization.

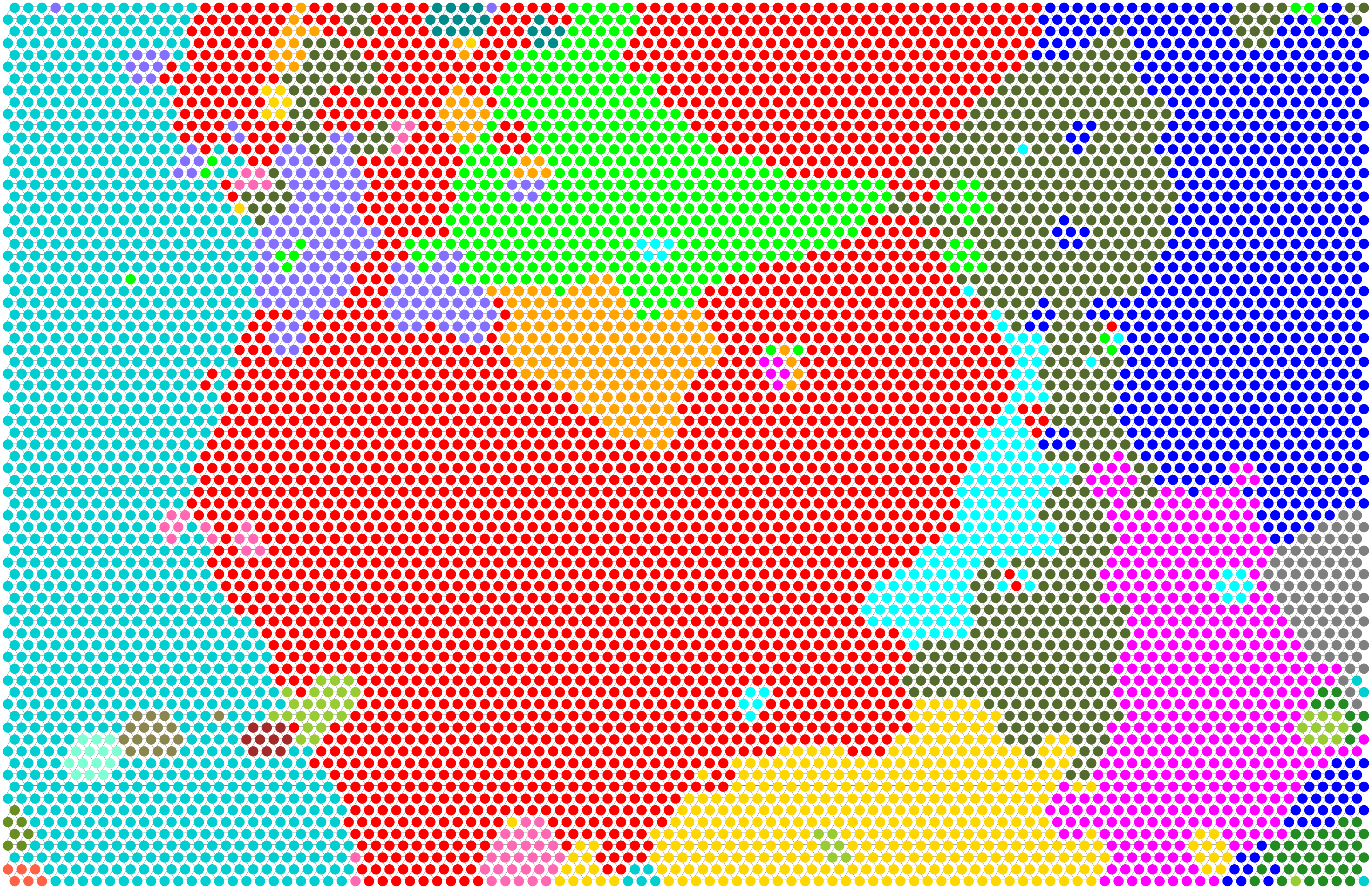


Figure 6.8: Psychometric `men` database visualized using the Self-Organizing Maps mapping: decision borders are visualized.

6.1.2 Detection of outliers

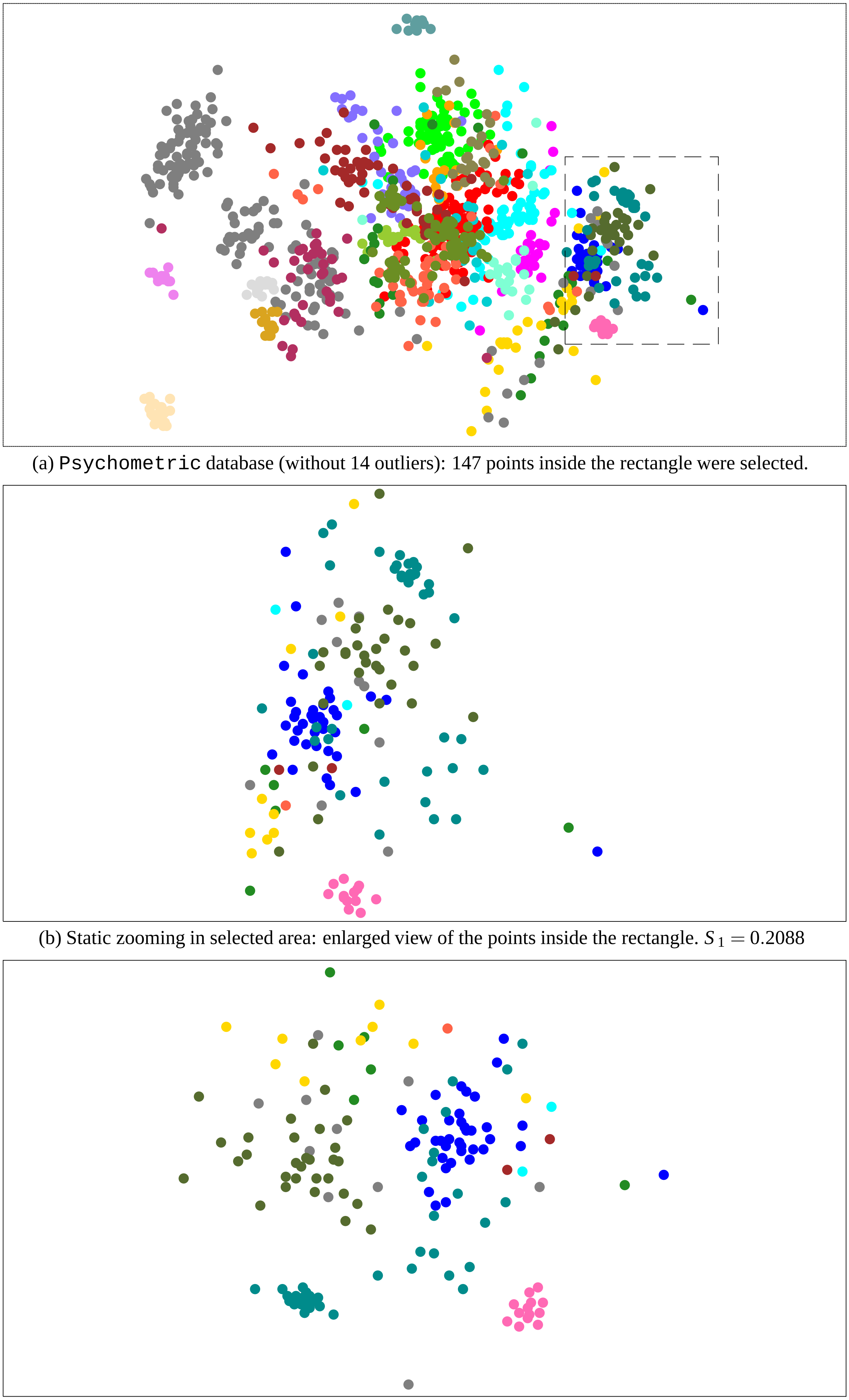
A second application of MDS to the visualization of medical databases is the detection of outliers. In general in statistics, outliers are atypical, infrequent observations. Due to various incidents that may occur during the data acquisition procedure (such as erroneous measurement or fatigue of the practitioner), some cases in the database may be wrongly labeled or located in the data space very far from the rest of the data. Such cases can be easily detected when visualizing the whole database cases in one display. The database generally forms a dense cluster of points whereas outliers are isolated points located outside of the main cluster. Examples of outliers can be seen in figure 6.2, in which one point appears isolated on the left side (point nb. 941 from class `dewiacyjny_styl_odpowiedzi_1_k`), as well as the two rightmost points (point nb. 357 from class `paranoja_w` and point nb. 29 `norma_w`) in figures 6.2 and 6.9. The fact that those two outliers are close from one another but belong to different classes suggests that one of those subjects may have been wrongly diagnosed and labeled. Wrongly labeled data can be detected as well if the different classes are naturally well separated, whereas some points appear in the neighborhood of points forming a cluster of a different class.

6.1.3 Zooming in interactively chosen data subspaces

When visualizing a large database, we often need to focus our view on particular data and its neighborhood. Viewing a part of a database that is chosen from the whole database display is called here “zooming”. It is possible to perform such a zoom by just enlarging the scale factor between the representation space $\mathcal{M}$ and the screen coordinates space and such tools were developed. This first zooming tool based on the whole database map is called *static* zoom because we always see portions of the same map with changing magnification factors. Another possibility which we call *dynamic* zooming is to select from the database display data points defining the data subspace to be viewed magnified, and then to perform again a mapping of the selected points separately by MDS. An interactive data points selection tool allowing to select some points directly from the plot was implemented in our software in this purpose. Different selection methods are available: select one point separately by clicking on it, select points in a rectangle, select the k nearest neighbors (in the map space $\mathcal{M}$ or in the data space $\mathcal{D}$) of a chosen point, or select the points within a radius r around a chosen point. It is possible to deselect part of the selected points using those methods.

The display of the selected points when mapped separately can be very different than when those points were mapped among thew other database points, due to the reduced number of distances constraints. Even if it is similar, it can appear rotated or reversed (upside-down or right-left). A Procrustes analysis could be applied here to rotate and possibly flip the zoomed data points configuration, which is not included in our software at its present stage of development. An example of zooming in a subspace of the psychometric `women-b`[2] database is shown in figure 6.9. We see that dynamic zoom configuration (fig. 6.9(c)) is quite different from the static one (fig. 6.9(b)), and its Stress value S_1 is much lower, so the view in figure 6.9(c) is topographically more correct.

[2] This is a different women database than the one visualized in §6.1.1. This second database is made of 1027 samples into 27 nosological classes.



(a) `Psychometric` database (without 14 outliers): 147 points inside the rectangle were selected.

(b) Static zooming in selected area: enlarged view of the points inside the rectangle. $S_1 = 0.2088$

(c) Dynamic zooming in selected area: new configuration after a separate mapping. $S_1 = 0.0514$

Figure 6.9: Zooming in an interactively chosen database subspace using MDS mapping.

6.2 Visualization of classifiers decision borders

So far we applied MDS to the visualization of data sets to get an insight into the different data classes by looking at their relative positions or distributions in the data space. We are also able to locate a new data on an existing map. It can be of great help for people facing the problem of the choice of the appropriate classifier to a given classification task to see not only the classes of the training data, but also how a given classifier separates them. As well as classes involved in a classification task can be represented by subsets of particular objects or points, a classifier's decision borders can be represented by points in the data space. The following notations will be used in this section:
$S_{TD} = \{P_{TD_i}, i = 1,...,N_{TD}\}$, the set of N_{TD} training data points in data space $\mathcal{D}$,
$S_{DB} = \{P_{DBi}, i = 1,...,N_{DB}\}$, the set of N_{DB} decision border points in data space $\mathcal{D}$,
$s_{TD} = \{p_{TD_i}, i = 1,...,N_{TD}\}$, the set of training data points in mapping space $\mathcal{M}$,
$s_{DB} = \{p_{DB_i}, i = 1,...,N_{DB}\}$, the set of decision border points in mapping space $\mathcal{M}$,

Classification tasks are most of the time based on a set of several independent features, so the space in which classification operates (also called here *feature* space) has as many dimensions as there are features. But the number of features really needed by the classifier to learn the training data (or to classify them well) may be smaller than the total number of available features; the interesting features can be selected in a pre-processing stage of *feature selection*. When a classifier uses only a reduced number of features, the mapping of the decision border should be based also on those selected features. This principle was applied for all experiment in this section. Multidimensional scaling can be applied to the visualization of the training data and decision borders of a classifier (both featured by multivariate data and represented by points on a plot). The heuristics that we designed to produce the decision borders points often lead to a great number of new multivariate data. As we wish to visualize those data on the same plot as data points representing the classes, the computation time and memory needed to map all those points together in one batch becomes prohibitive (for the reasons given in §4.3.1). A solution to this problem is to apply the relative mapping in the following way:

1. Use standard MDS to map only the classifier's training data points $\{P_{TDi}\}$,

2. Use relative MDS to map only the classifier's decision borders points $\{P_{DBi}\}$ with respect to the mapped training data points $\{P_{TDi}\}$, that are now fixed. This second step can be performed in three ways: a) If the number N_{TD} of generated decision borders points is small enough, then map points $\{P_{DBi}\}$ and $\{P_{TDi}\}$ together using standard MDS. b) If N_{TD} is too large to allow a 1 run mapping, then map points $\{P_{DBi}\}$ using relative MDS mapping in one batch, or c) map points $\{P_{DBi}\}$ using relative MDS mapping sequentially in N_{DB} batches of 1 point each.

In the case of small data sets, we visualized the decision border points together with the training data points by mapping them using MDS in 3 possible manners: a) all training data and border points mapped together in one MDS run, b) all training data firstly mapped by MDS, all border points $\{P_{DBi}\}$ mapped in one MDS run relative to the data points, and c) all training data firstly mapped by MDS, each border point P_{DBi} is mapped relative to the data sequentially. The purpose was to find out how much the choice of the technique used for the mapping influences the resulting configuration. The mappings obtained for the `appendicitis` data set (see §6.2.1) together with several decision border sets showed that the 3 methods lead to very similar configurations.

The problem of visualizing a classifier's decision borders can be approached in two quite different ways. The first one is *generative* because the border itself is shown, the second one is *inductive* because the border is not directly shown but induced by the resulting display. Those two approaches are defined here below:

- **Approach 1**: Generate new points that are located on the decision borders and map them onto the map of the training data (the decision border is made by the "line" connecting the new points),
- **Approach 2**: Generate new points in the neighborhood of the decision borders, classify them and map them onto the map of the training data (the decision border is seen as the interface of different areas filled and colored by the new points),

6.2.1 Visualization of classifiers decision borders – Approach 1

Three different heuristics have been designed in order to generate new points in the feature space that are located on a classifier's decision border:

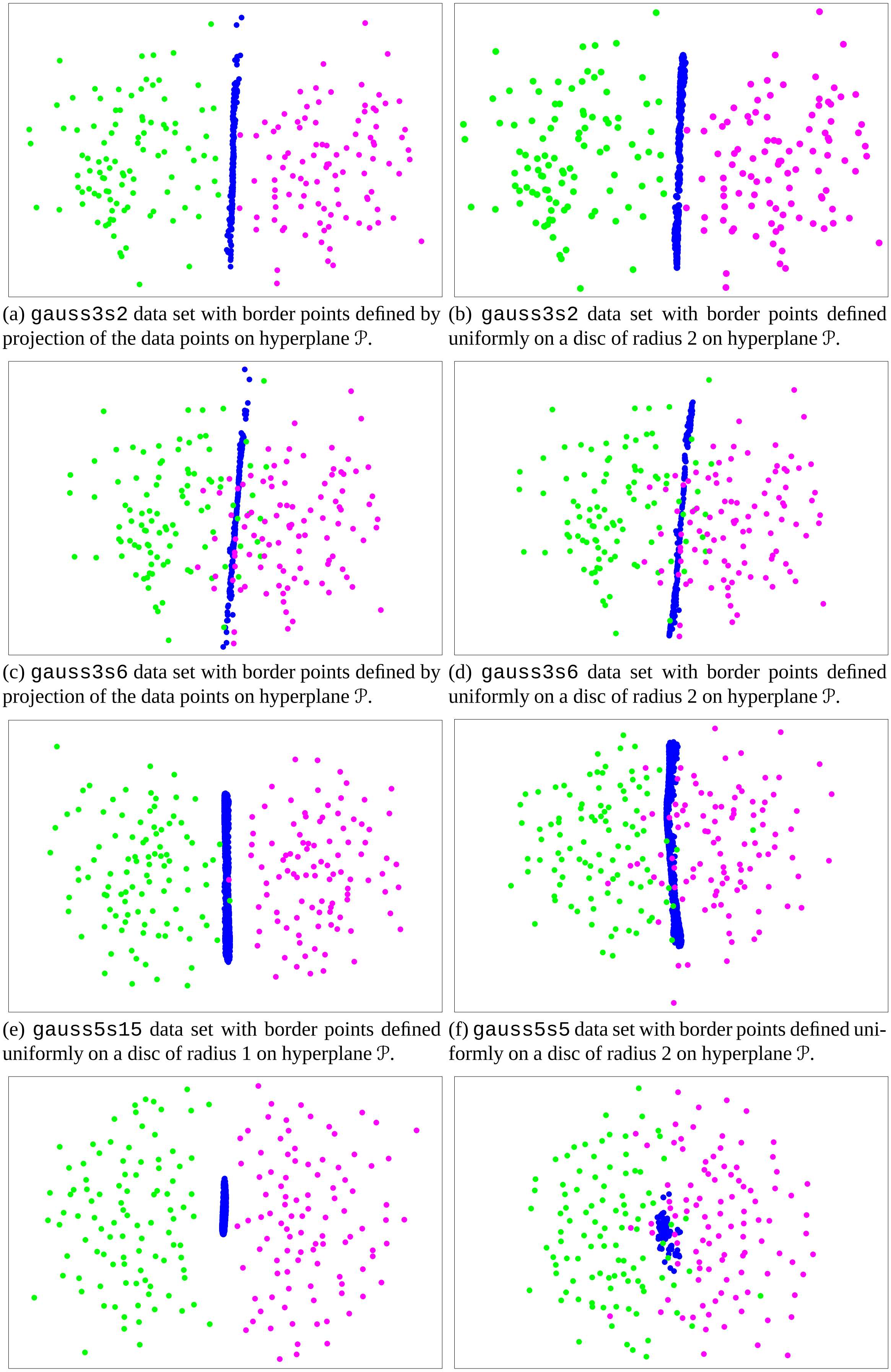
1. Generate new points directly on decision borders hyperplanes,
2. Generate new points on lines between data points,
3. Generate new points on lines between vertices of a discretized data space.

Generate new points on decision borders hyperplanes

In the case of a classifier based on rules, the decision borders are defined in the feature space by hyperplanes that are analytically known. Decision borders points can be found whether by perpendicular projections of the training data on these hyperplanes, or by random generation of points from the hyperplanes. In this last case, the points are generated from a uniform distribution, in the neighborhood of the data and not anywhere on the hyperplane. In a purpose of illustration, we designed artificial data sets in data spaces with $D = 3$, $D = 5$ and $D = 10$ dimensions. Each of these data sets is made of two clouds of 100 points formed using a Gaussian distribution[3] centered at points $P_1(x_{11} = -1, x_{1i} = 0, i = 2, ..., D)$ and $P_2(x_{21} = +1, x_{2i} = 0, i = 2, ..., D)$. The Gaussian distributions have null covariances and equal variances in all dimensions: We chose i) $\sigma = 0.2$ to have 2 well separated classes, and ii) $\sigma = 0.6$ to have a bit of overlapping between the two classes. The ensuing data sets are called `gauss3s2` ($D = 3$ and $\sigma = 0.2$), `gauss3s6` ($D = 3$ and $\sigma = 0.6$), and similarly `gauss5s15`, `gauss5s5`, `gauss10s2` and `gauss10s5`. The hyperplane defined by $\mathcal{P} = \{(x_1, ..., x_D)^T : x_i = 0, i = 1, 4, ..., D\}$ is theoretically the optimal[4] hyperplane separating those two classes, so it can be regarded as a decision border hyperplane. Then we generated 200 decision border points on hyperplane $\mathcal{P}$ in the following two ways: a) project perpendicularly the points from the two classes on $\mathcal{P}$, b) generate random points on $\mathcal{P}$ with a uniform distribution limited to the area between the Gaussian distributions, that is on a disc centered at the origin with radius such that the decision border is as long as the data range. The two Gaussian clouds were first mapped using MDS and the decision border points were added by relative MDS, the resulting maps are shown in figure 6.10.

[3] Gaussian distributions were generated using the `RANLIB` library available at the Statlib repository.

[4] It would be optimal in the case of infinite number of points generated in the Gaussian distributions.



(a) `gauss3s2` data set with border points defined by projection of the data points on hyperplane $\mathcal{P}$.

(b) `gauss3s2` data set with border points defined uniformly on a disc of radius 2 on hyperplane $\mathcal{P}$.

(c) `gauss3s6` data set with border points defined by projection of the data points on hyperplane $\mathcal{P}$.

(d) `gauss3s6` data set with border points defined uniformly on a disc of radius 2 on hyperplane $\mathcal{P}$.

(e) `gauss5s15` data set with border points defined uniformly on a disc of radius 1 on hyperplane $\mathcal{P}$.

(f) `gauss5s5` data set with border points defined uniformly on a disc of radius 2 on hyperplane $\mathcal{P}$.

(g) `gauss10s2` data set with border points defined uniformly on a disc of radius 1 on hyperplane $\mathcal{P}$.

(h) `gauss10s5` data set with border points defined uniformly on a disc of radius 2 on hyperplane $\mathcal{P}$.

Figure 6.10: Two multivariate Gaussian distributions with a planar decision border.

As it could be expected, the hyperplane $\mathcal{P}$ separating the 3-dimensional Gaussian distributions is represented by straight lines on the four displays of (figures 6.10(a) to 6.10(d)). For the 5-dimensional Gaussian distributions (figures 6.10(e) and 6.10(f)), we observe that the border line gets thicker and is shorter. This effect is more clearly visible on the plots of the 10-dimensional Gaussian distributions (figures 6.10(g) and 6.10(h)), where the hyperplane is reduced to a cluster in the midway between the two Gaussians. Those distortions are involved by the greater reduction of dimensionality (from 10 or 5 to 2) than in the previous cases (from 3 to 2). The distortions result from the fact that when an hyperplane embedded in a higher dimensional space is defined by $\{(x_1,...,x_D)^T : x_i = 0, i = 1,4,...,D\}$, all the dimensions other than x_2 and x_3 collapse to one point, hence the resulting central cluster. From this last observations we are inclined to think that in high dimensional real data sets, decision borders (even simple 2-dimensional ones, that is defined as hyperplanes) may be difficult to see as separate straight lines on a planar display.

Generation of new points on lines between data points

The idea to find a point on the decision border between two classes is to move a point in the feature space on a straight line between two data points belonging to different classes and to check the class of the moving point as attributed by the classifier. When its class changes, it means that we moved over the decision border and this point is kept as a decision border point. We generate decision border points on lines between data points from two classes as follows:

- select a subset $S'_{TD} = \{P'_{TDi}, i = 1,...,N'_{TD}\}$ of the training data set S_{TD} containing those points that are close to the decision border: these are the training data points around which in a given neighborhood there are training data points from a different class. This neighborhood can be defined by a given radius or by a given number of nearest neighbors. An alternative is to take all the points in S_{TD} if they are not too numerous.

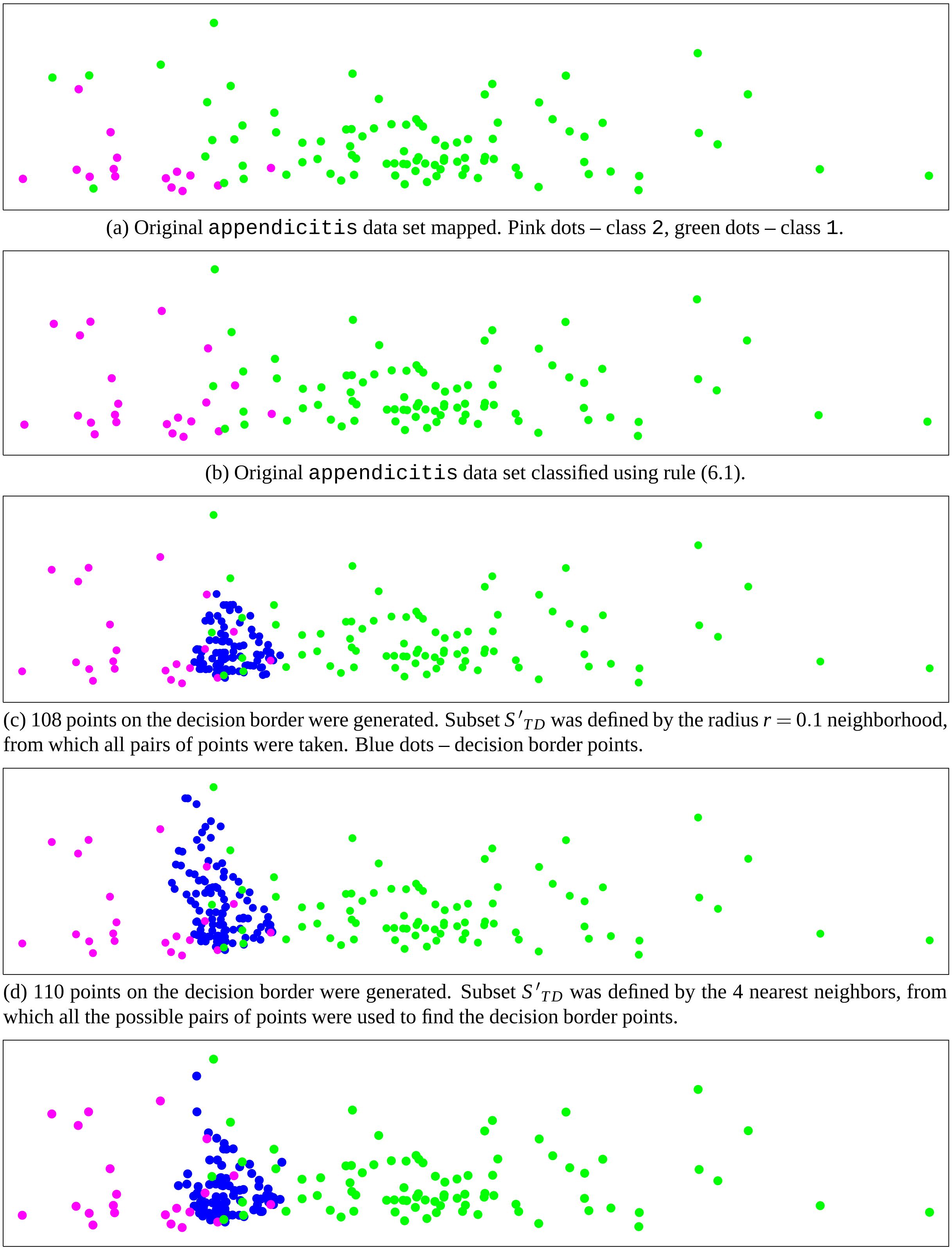
- select pairs of points $\{(P_1,P_2) : C(P_1) \neq C(P_2)\}$ from subset S'_{TD}, such that the two points belong to different classes: the pairs are formed by neighboring points, whether within a hypersphere of given radius r or among the k nearest neighbors,

- find the border points following the scheme: for each selected pair of points (P_1,P_2), move a point P_{DB} on the line joining P_1 to P_2 (stepping by a small value ε) checking its class $C(P_{DB})$, and stop when $C(P_{DB})$ changes from $C(P_1)$ to $C(P_2)$. Add point P_{DB} to the decision border set S_{DB}.

This technique was applied to the visualization of decision borders of the `appendicitis` data set. This data set provided by Shalom Weiss from Rutgers University contains 106 vectors with 8 numerical attributes (named WBC1, MNEP, MNEA, MBAP, MBAA, HNEP and HNEA), distributed into 2 classes: 85 cases belong to class “`1`” (80,2% of the data) are severe appendicitis cases and 21 cases belong to class “`2`” (19,8%) that are other disease cases. The visualized classification rule is:

$$\texttt{if } (MNEA \geq 6650) \texttt{ then } class\_1 \texttt{ else } class\_2. \tag{6.1}$$

The decision border of this rule is defined in the data space by an hyperplane, but as can be seen in figure 6.11 the points representing the rule are not mapped on a straight line due to distortions of the data topography.



(a) Original `appendicitis` data set mapped. Pink dots – class `2`, green dots – class `1`.

(b) Original `appendicitis` data set classified using rule (6.1).

(c) 108 points on the decision border were generated. Subset S'_{TD} was defined by the radius $r = 0.1$ neighborhood, from which all pairs of points were taken. Blue dots – decision border points.

(d) 110 points on the decision border were generated. Subset S'_{TD} was defined by the 4 nearest neighbors, from which all the possible pairs of points were used to find the decision border points.

(e) 107 points on the decision border were generated. Subset S'_{TD} was defined by the 10 nearest neighbors, from which only the pairs of points within a radius $r = 0.8$ neighborhood were taken.

Figure 6.11: Visualization of `appendicitis` data set with classification rule (6.1).

Generation of new points on lines between vertices of a discretized data space

The technique presented in this section is similar to the one of the previous section, with the difference that instead of searching the decision border points on the basis of the training data points, we use new points. Those points are defined e.g. on a regular lattice that "fills" the feature space, then their classes are found using the classifier. The method to generate new points on lines between vertices of such a discretized data space is as follows:

- First generate a regular lattice of points L that covers completely the area of the feature space containing data,
- Second, proceed as in previous section but using the vertices of the lattice L instead of the training data set S_{TD}, i.e.:
  - select the vertices of L that are close to the decision borders,
  - make pairs of selected points that belong to different classes: $\{(P_1, P_2) : C(P_1) \neq C(P_2)\}$,
  - find the border point by: moving a point P_{DB} from P_1 to P_2 (stepping by a small value ε), and stop when the class of P_{DB} changes from $C(P_1)$ to $C(P_2)$. Add point P_{DB} to the decision border set S_{DB}.

Initialization of decision border points for the relative MDS mapping

The points $\{p_{DB}\}$ representing in the mapping space the decision border points $\{P_{DB}\}$ can be initialized before the MDS mapping by a linear interpolation technique. The idea is to compute the relative position in the data space of point P_{DB} between the two points P_1 and P_2 from which it was found, and to transfer this ratio to the mapping space to place $\{p_{DB}\}$ between p^1_{DB} and p^2_{DB}. In vector notation, this gives:

$$\overrightarrow{p^1_{TD}p_{DB}} = \frac{\left\|\overrightarrow{P^1_{TD}P_{DB}}\right\|}{\left\|\overrightarrow{P_{DB}P^2_{TD}}\right\|} \cdot \overrightarrow{p_{DB}p^2_{TD}} \tag{6.2}$$

If the decision border points are mapped individually, then only one point is moved during each minimization run. Then the number of local minima of the Stress function is very small (there will be probably a few local minima, as shown in figure 4.6), so that the choice for the initial position of the mapped point is not crucial.[5]

These heuristics were primarily designed to the visualization of data sets made of two classes only, and with a decision border presumably neither strongly bent nor high dimensional (generally with a quite linear or planar shape). Different other heuristics can be proposed in the cases where there are more than two classes or with more strongly bent borders, leading to a varying number of decision border points with respect to the number of data points.

6.2.2 Visualization of classifiers decision borders – Approach 2

In this approach, the points generated in order to visualize a classifier's decision borders need to be placed *in the neighborhood of* and not *on* the decision borders. The decision border will be visualized not by the generated points themselves, but it will be visible as the interstice between areas filled in distinct ways (e.g. different colors if the point are marked by colored dots). The technique used to find those points is the following:

[5] When the number of fixed points is not too large (about 100), we obtain very often exactly the same final positions after linear interpolation or random initializations, but convergence is faster after linear interpolation.

- first select a subset of the database if it is very large in order to reduce the number (and the complexity) of the decision borders visualized,

- select a subset $S'_{TD} = \{P'_{TDi}, i = 1, ..., N'_{TD}\}$ of the training data points containing those that are close to the decision border: these are the training data points around which in a given neighborhood there are training data points from a different class. This neighborhood can be defined by a given radius or by a given number of nearest neighbors.

- around each selected point P'_{TDi}, generate a given number N_G of points from a Gaussian distribution centered at P'_{TDi} and with equal variances in all dimensions (the value of σ should be chosen so that the subspace containing the selected points is uniformly filled with new points),

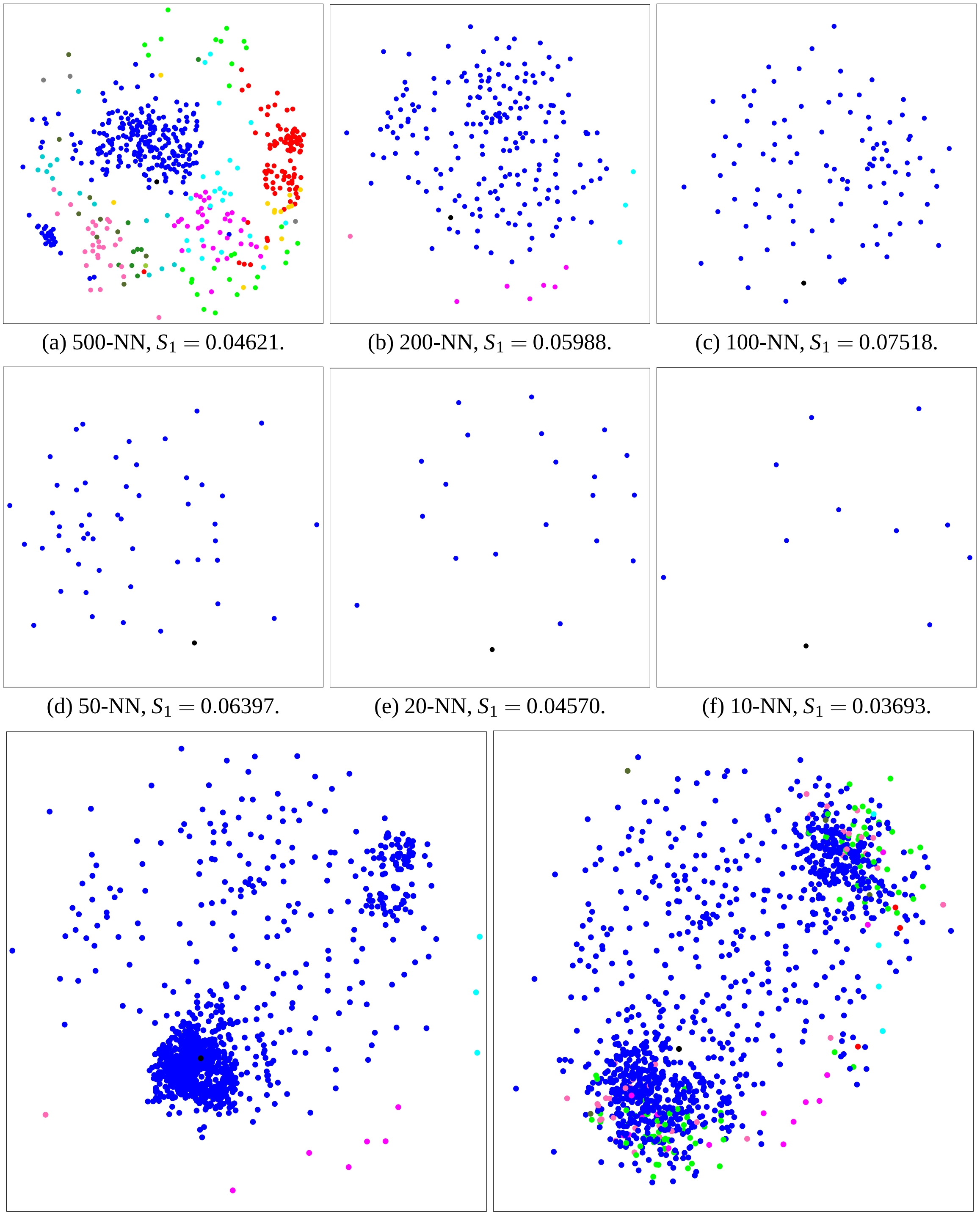
- use the classifier to label all the points generated in the Gaussian distributions.

This approach was applied to the visualization of decision borders of the classifier called Incremental Network (`IncNet`) [72] trained on the `women psychometric` database, on which this classifier performs very well (only 8 errors on the 1711 database cases, i.e. 99.53 % correct classification). It is however interesting to understand why this classifier fails on a few cases, and to correlate `IncNet`'s classification probabilities for the proposed and alternative classes with the distributions of the classes in the data points neighborhoods. A few number of data points were selected, as well classified with a high level of certainty (points with indices 5, 270 and 554) and points detected as particularly difficult to learn, and consequently to classify correctly (points with indices 426 and 604). MDS mapping has been used to zoom into the neighborhood of these data points, and then to visualize the database's classes distribution in these neighborhoods. Class assignments by `IncNet` for selected data points and the corresponding probabilities are presented in Table 6.1.

Data index	True class	Assigned class	(probability)	Alternative class	(probability)
5	norma	norma	(0.???)	???	(0.???)
554	organika	organika	(0.826)	schizofrenia	(0.062)
604	organika	schizofrenia	(0.544)	organika	(0.436)
270	nerwica	nerwica	(0.921)	schizofrenia	(0.042)
426	nerwica	schizofrenia	(0.428)	nerwica	(0.309)

Table 6.1: Classification by neural network (`IncNet`) of chosen data points from the `psychometric` database.

Neighborhoods and classes distributions for the corresponding data points are shown in the following figures 6.12 to 6.16. The colors used to represent the data classes are according to the legend of figure 6.2 on page 60.



(a) 500-NN, $S_1 = 0.04621$.

(b) 200-NN, $S_1 = 0.05988$.

(c) 100-NN, $S_1 = 0.07518$.

(d) 50-NN, $S_1 = 0.06397$.

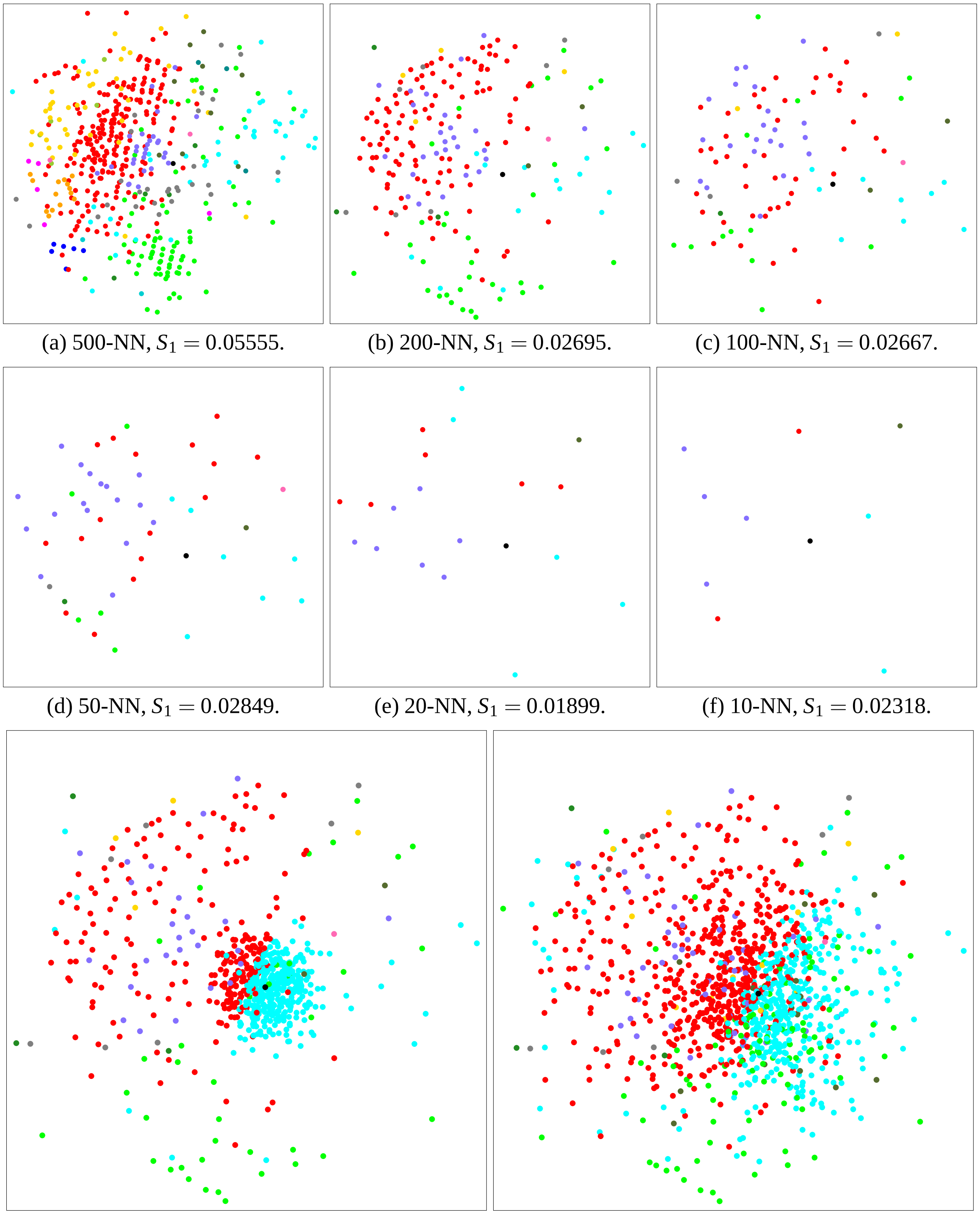
(e) 20-NN, $S_1 = 0.04570$.

(f) 10-NN, $S_1 = 0.03693$.

(g) 200-NN + 1000 Gaussian points ($\sigma = 1$).

(h) 200-NN + 1000 Gaussian points ($\sigma = 10$).

Figure 6.12: Zooming in the neighborhood of data `p5` (black dot) from class `norma` (norma–blue, schizofrenia–red, nerwica–green) on plots a to f. `IncNet` classifier's decision borders on plots g and h.



(a) 500-NN, $S_1 = 0.05555$.

(b) 200-NN, $S_1 = 0.02695$.

(c) 100-NN, $S_1 = 0.02667$.

(d) 50-NN, $S_1 = 0.02849$.

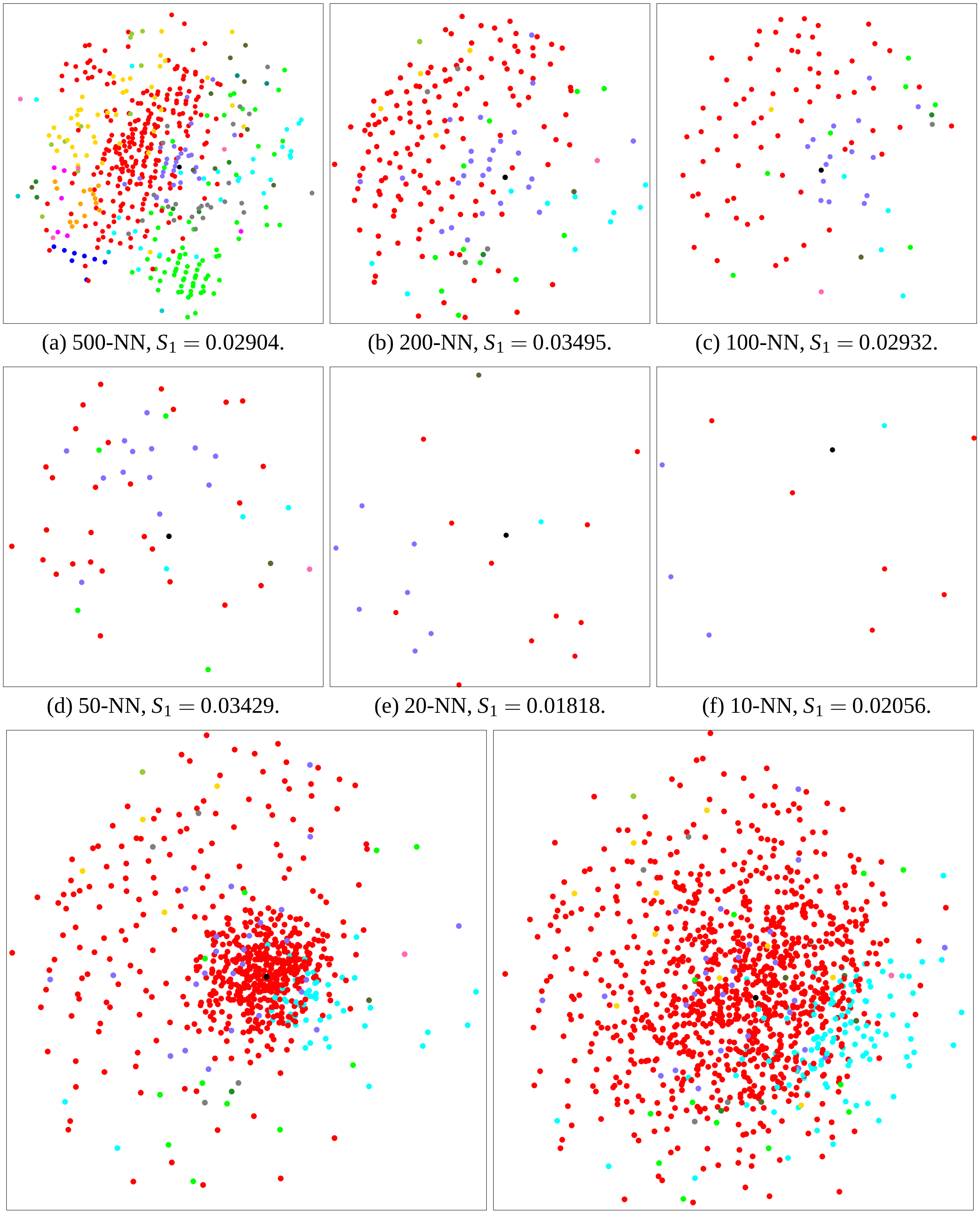
(e) 20-NN, $S_1 = 0.01899$.

(f) 10-NN, $S_1 = 0.02318$.

(g) 200-NN + 500 Gaussian points ($\sigma = 1$).

(h) 200-NN + 1000 Gaussian points ($\sigma = 4$).

Figure 6.13: Zooming in the neighborhood of data `p554` (black dot) from class `organika` (organika–light blue, schizofrenia–red, nerwica–green) on plots a to f. `IncNet` classifier's decision borders on plots g and h.



(a) 500-NN, $S_1 = 0.02904$.

(b) 200-NN, $S_1 = 0.03495$.

(c) 100-NN, $S_1 = 0.02932$.

(d) 50-NN, $S_1 = 0.03429$.

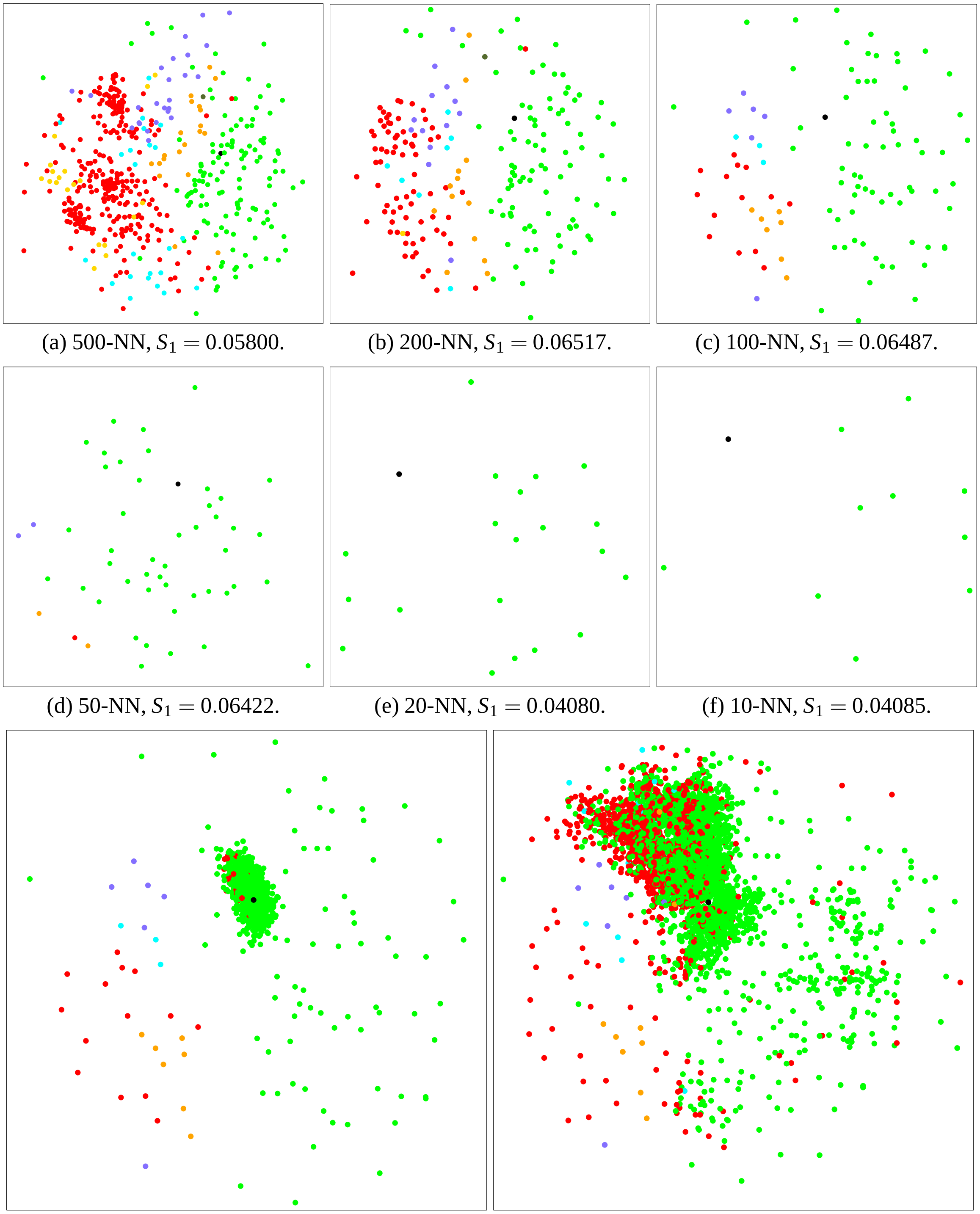
(e) 20-NN, $S_1 = 0.01818$.

(f) 10-NN, $S_1 = 0.02056$.

(g) 200-NN + 500 Gaussian points ($\sigma = 1$).

(h) 200-NN + 1000 Gaussian points ($\sigma = 4$).

Figure 6.14: Zooming in the neighborhood of data `p604` (black dot) from class `organika` (light blue) on plots a to f. `IncNet` classifier's decision borders on plots g and h.



(a) 500-NN, $S_1 = 0.05800$.

(b) 200-NN, $S_1 = 0.06517$.

(c) 100-NN, $S_1 = 0.06487$.

(d) 50-NN, $S_1 = 0.06422$.

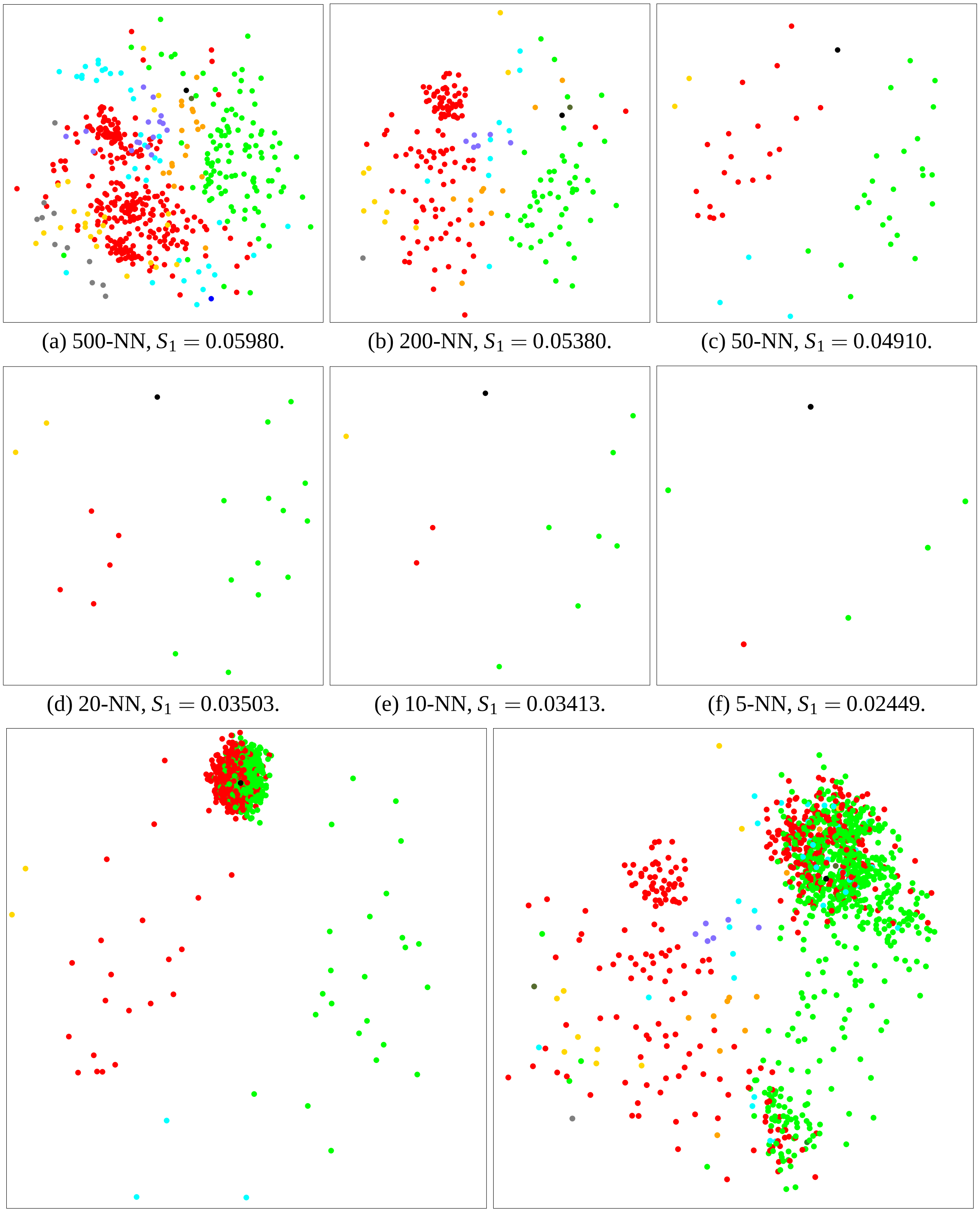
(e) 20-NN, $S_1 = 0.04080$.

(f) 10-NN, $S_1 = 0.04085$.

(g) 100-NN + 1000 Gaussian points ($\sigma = 1$).

(h) 100-NN + 5000 Gaussian points ($\sigma = 8$).

Figure 6.15: Zooming in the neighborhood of data `p270` (black dot) from class `nerwica` (green) on plots a to f. `IncNet` classifier's decision borders on plots g and h.



(a) 500-NN, $S_1 = 0.05980$.

(b) 200-NN, $S_1 = 0.05380$.

(c) 50-NN, $S_1 = 0.04910$.

(d) 20-NN, $S_1 = 0.03503$.

(e) 10-NN, $S_1 = 0.03413$.

(f) 5-NN, $S_1 = 0.02449$.

(g) 50-NN + 1000 Gaussian points ($\sigma = 1$).

(h) 200-NN + 1000 Gaussian points ($\sigma = 8$).

Figure 6.16: Zooming in the neighborhood of data `p426` (black dot) from class `nerwica` (green) on plots a to f. `IncNet` classifier's decision borders on plots g and h.

Discussion

- **Zooming views** The views representing the progressive zoomings towards the chosen data points (figures (a) to (f)) have been manually rotated and flipped when necessary in order to exhibit continuity during the zooming process. Hence the zooming in the neighborhood of the central black point appears as a continuous process for each of the five studied points. When taking less and less points in the neighborhood of the central point (from 500 to 10), the evolution of the final Stress value S_1 is not necessarily decreasing. The final Stress values are rather related to how much the clouds of mapped points can be embedded in a plane without too much distortions. For example, it can happen that 10 points form roughly a 9–D manifold (as in a simplex) that will be difficult to represent correctly on a plane, but the same 10 points surrounded by 100 other points form globally a manifold that can be more easily embedded in a 2–D space (if the 100 points do lie on a plane). Following Kruskal's guidelines[6] [89], most of the mappings we obtained are satisfying representations of the real 13–dimensional neighborhood relationships. From the observed Stress values, we conclude e.g. that the figures zooming the neighborhood of point `p604` are much more reliable than the figures of the neighborhood of point `p270`.

- **Decision borders views** For each studied point `p_i`, we generated sets of N_n new points drawn from multivariate (13–dimensional) Gaussian distributions centred at point `p_i` with diagonal covariance matrix and σ_i values identical in all dimensions. We chose to build several new sets with parameters $(N_n = 500, \sigma_i = 1)$, $(N_n = 1000, \sigma_i = 4)$ and $(N_n = 5000, \sigma_i = 8)$, in order to produce areas of different widths. We expected to find the new data on discs centered at point `p_i`, with classes separated according to the distribution of the surrounding basis points. A first observation of the resulting displays is that the smaller new sets are more concentrated than the larger ones, this was to be expected. A second conclusion is that when the new points are classified into various classes, the classes are better separated on the displays for the studied points zoomed in with low final Stress values. This can be explained by the fact that the new points are added to less constrained configurations and so they find more easily a good minimum. The larger new sets are not well represented on the displays which zooms had high final Stress values. This can be seen on figures 6.12(h) and 6.15(h) where the new mapped points are distributed in different places, and not all centered around point `p_i`, probably due to the presence of local basins of attractions on those displays.

  It is interesting to try to correlate, for each analyzed data point, `IncNet`'s classification results (and the corresponding probabilities shown in Table 6.1) to the class areas displayed around the points on the local zooms. Point `p554` is assigned by `IncNet` to class `organika` with high probability, which is confirmed by the neighborhoods in figures 6.13 (e and f), where the black dot lies among blue dots representing class `organika`. Point `p604` is assigned by `IncNet` to class `schizofrenia` with a relatively low probability, this can be seen in figures 6.14 (e and f) because the black dot is surrounded by red dots representing class `schizofrenia` mixed with blue ones. Similarly, the affectation of point `p270` (resp. `p426`) to class `nerwica` (resp. `schizofrenia`) is confirmed by plot 6.15(e) (resp. 6.16(e)) in which the black dot lies among green (resp. red) ones.

[6] From the experience of the authors, a final Stress value $S_1 \leq 0.01$ is considered to be good (the configuration is reliable), a final Stress $0.01 \leq S_1 \leq 0.05$ is satisfying and $S_1 \geq 0.05$ is poor.

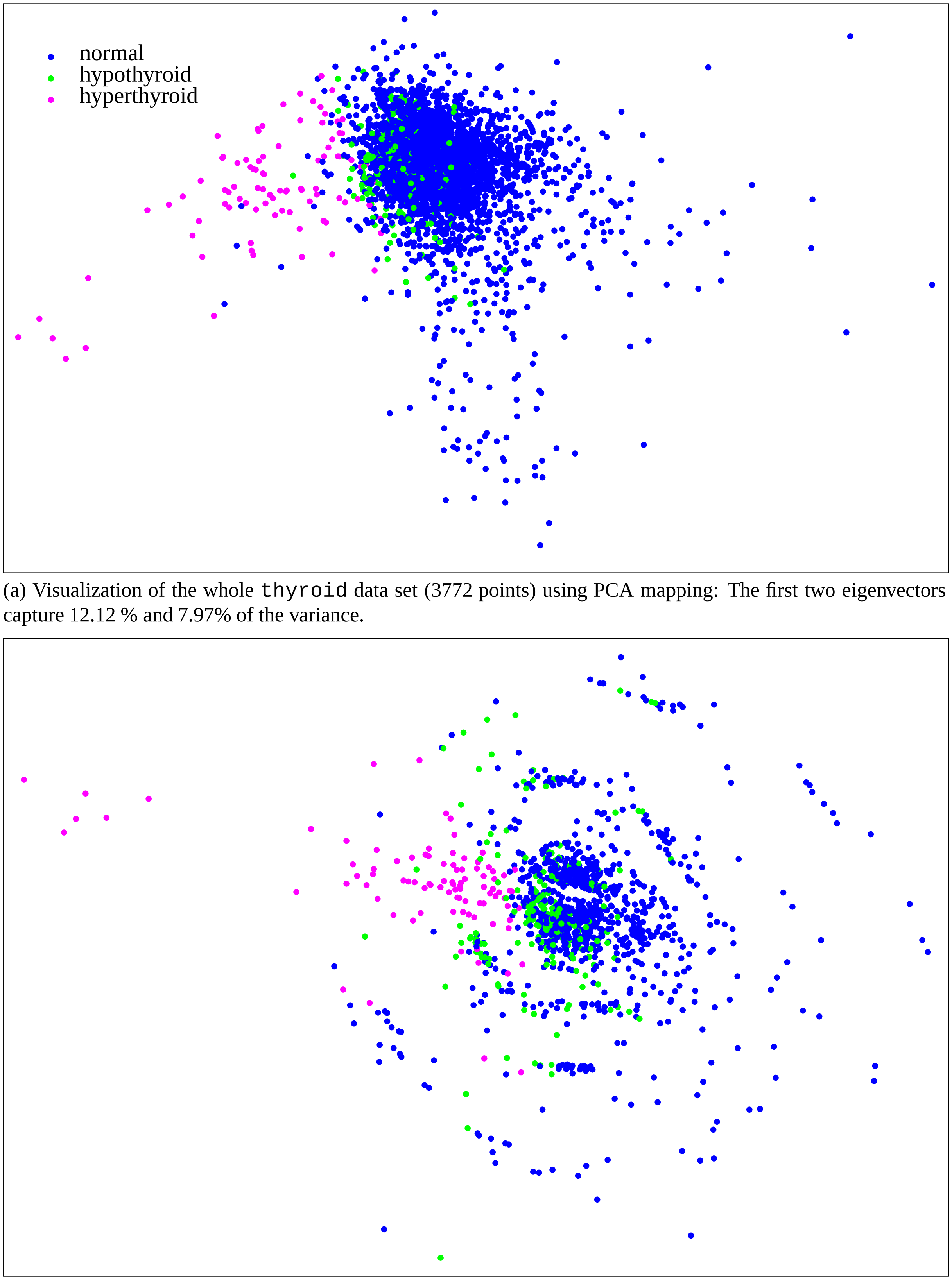
6.2.3 Conclusion

The different maps we obtained by the proposed procedures show that high dimensional decision borders visualization is a difficult task. Multidimensional scaling exhibited a good visualization capability in the case of planar decision borders. In the more general case of non linear decision borders, or when they are embedded in high dimensional data space, the mapping results are not always satisfying, depending on how much the decision border must be distorted to be embedded in a plane. It can be advanced that a clear and faithful (that is, with the classes well separated) 2–dimensional representation of a high dimensional surface with respect to a high dimensional data set is impossible in the general case. Nevertheless, the obtained displays convey some information useful to understand a classifier's behavior in an observed region. The number of classes represented and the amount of between class overlap on the display give an insight on the classifier's discriminancy in the area under consideration. The width of the displayed decision border, when it is observable, also gives an indication on how much confident one can be about the quality of the representation of the classes in the considered region, as well as the final Stress value.

6.3 Visualization of other medical data

6.3.1 Visualization of large data sets

In the context of our work, a large data set is one for which the number of cases or objects does not allow a direct visualization using MDS mapping, say more than 1500 cases by present computer memory capacities. The `thyroid` data set is a public domain data set that has been made available at the UCI repository by Randolf Werner from Daimler-Benz. It contains 3772 cases with 21 attributes, 15 attributes are binary and 6 are continuous. The problem is to determine whether a patient referred to the clinic is hypothyroid. Therefore three classes are built: `normal` (not hypothyroid, 3488 patients), hyperfunction (`hyperthyroid`, 191 patients) and subnormal functioning (`hypothyroid`, 93 patients). The number of cases does not allow a mapping by MDS in one run. In order to visualize such a data set, several strategies are possible: i) Select a subset following some heuristic, map this subset using standard MDS and add to it the remaining data points using relative MDS; ii) Reduce the data set's size using a clustering technique and map by standard MDS the cluster centers, to which the data can be added by relative MDS. We chose to reduce the data set's size by a point's selection technique. The question of whether to standardize the data or not is quite difficult to answer because of the presence of both binary (0/1) and numerical features (with range different from 1). Firstly, it comes to mind to standardize the data to set for all the features equal standard deviation, but it appeared that some binary features have almost all cases valued to zero, except a few ones. Standardization changes the ones to very high values and creates in this way artificial outliers. Normalization ... The heuristic to select data was guided by our interest in the visualization of the decision borders: We selected the points supposed to be close to the decision borders as follows: keep those points that have in their close neighborhood points belonging to another class than their own class, and delete the other points. The data set was hence reduced using the following scheme: For each data point $P_i, i = 1, \ldots, N$, if each of the k nearest neighbors of P_i belong to the same class as point P_i, then point P_i is removed from the data set. This technique applied for $k = 4$ allowed to reduce the number of data points in class `normal` from 3488 to 910, keeping all the points in classes `hypothyroid` and `hyperthyroid` unchanged. Displays of the `thyroid` data set are shown in figure 6.17.



(a) Visualization of the whole `thyroid` data set (3772 points) using PCA mapping: The first two eigenvectors capture 12.12 % and 7.97% of the variance.

(b) Visualization of the reduced `thyroid` data set (1194 points) using MDS mapping: PCA initialization ($S_1 =$ 0.3981), after 1000 iterations, $S_1 = 0,0502$. The circular traces drawn by points from class `normal` may be due to the 15 original binary features involving many identical inter-points distances (As for `simplex` data set).

Figure 6.17: Visualizations of `thyroid` data set: the number of points was reduced from 3772 to 1194 (2578 points from class `normal` that have their 4 nearest neighbors in class `normal` were removed from the data set).

6.3.2 Understanding classification results using the k–nearest neighbors classifier

In this section, we want to check visually the data points that are wrongly classified by the k–nearest neighbors (k–NN) classifier. The data set named `cukrzyca`, provided by prof. Szczepaniuk from the hospital in Łódź, describes 107 diabetic patients classified into two classes corresponding to two kinds of diabetes disease. Each of the 107 cases is featured with 12 attributes, there is no missing value. Among the 12 features, three are continuous and the 9 other features are binary. It was decided to set to 1 for `true` and 2 for `false`. A preprocessing of feature selection was performed on the data to obtain a final classification result as high as possible. This lead to retain only the 3 features with number 2, 3 and 9. A look at the 107 data points in those 3 dimensions shows that for many of them (40 points), the 3 retained features are identical. This is not desirable for the MDS algorithm because the Stress gradient is not defined then due to null distances. To face this difficulty, one can whether leave only one representative in each group of identical points (possible if all the identical points belong to the same class), or slightly move the points within the groups so that they become distinct. This second solution was preferred in this case because there were identical points belonging to different classes. The identical points were moved randomly within a radius of 0.2 around their original positions, in order to make them well separated on the plot shown in figure 6.18. It can be seen that the points are placed quite regularly following a grid. This effect is to be related to the discrete nature of most of the features. The very low value of final Stress $S_1 = 2.46e-4$ allows us to assume that the neighborhood relationships are well preserved. Each point wrongly classified by the k–NN classifier (for $k = 8$) is surrounded by a red circle. For each such point, it is possible to count the number of nearest neighbors from the same class (within the k nearest neighbors) and check that those points have a majority of neighbors from the other class. This allows to “see” why those points are wrongly classified by the k–NN classifier.

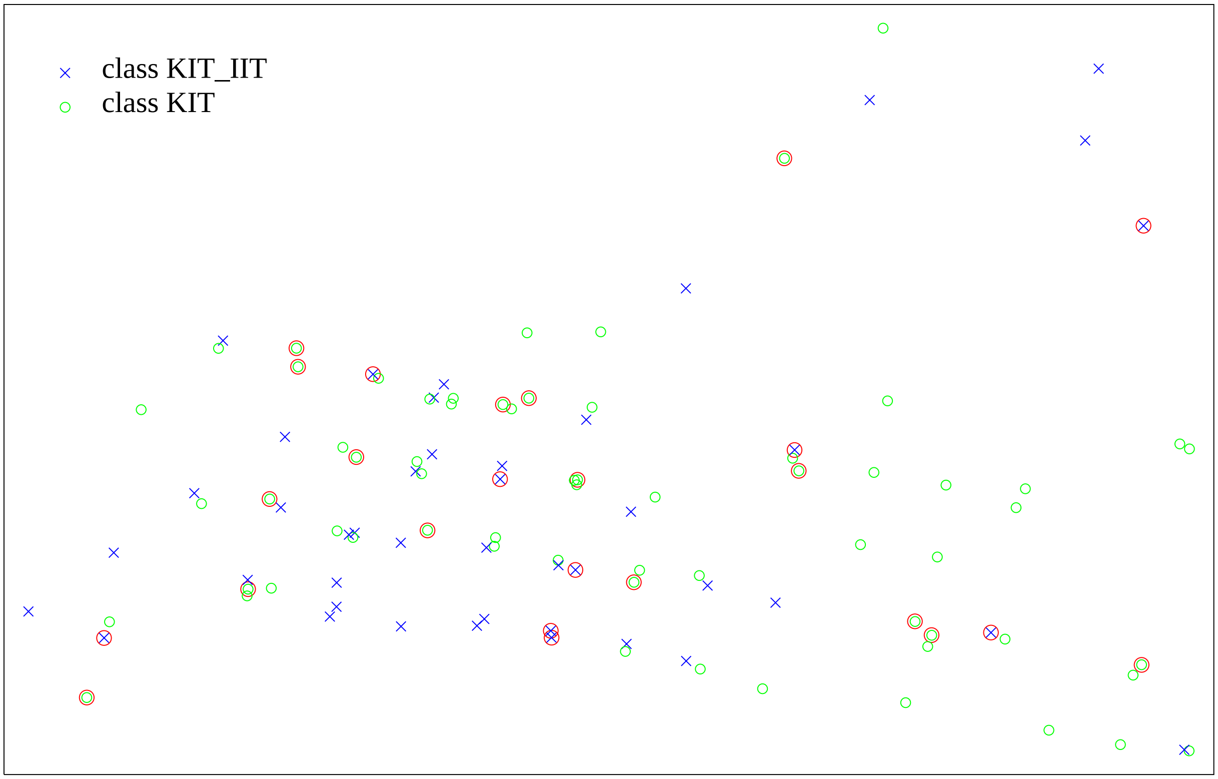


Figure 6.18: The `cukrzyca` data set: $S_1 = 2.46e-4$, 42-nd randomly initialized trial.

Chapter 7

Conclusions

7.1 Summary

The main objective of the thesis was to compare two mapping methods from the point of view of data topology preservation. In this purpose, the mapping methods have been tested on numerous artificial and real life data of different nature. It has been shown that Multidimensional Scaling is to be preferred to Self-Organizing Maps in this matter, despite its limitations. Improvements designed to face these limitations were proposed and tested, especially in the domain of Stress minimization. The problem of finding a reliable solution (by a global minimization of Stress function) is still crucial and should be investigated in further work. The Stress function was expressed in a framework general enough to include different approaches. The solutions retained to enable the visualization of large data sets by MDS (batch mapping and relative mapping) pushed away the memory requirements and computation time limitations. The increasing memory and calculation capabilities of computers should also make those limitations less and less significant. The interactive software `IMDS` allowed exploration of multivariate data sets and visualization of classifiers decision borders.

7.2 Further developments

In order to enhance the performances and usability of the data visualization software that we developed, some additional features to the MDS algorithm can be added:

Adapting weights on the distances in MDS

Depending on the purpose of the visualization of a particular data set, it can be helpful to have the possibility to adjust additional weight values defined for each inter-point distance in order to force its preservation during mapping.

Visualization of the Minimal Spanning Tree

A further technique which may be useful in making a judgement between different mappings is that of plotting the links of the *minimum spanning tree* computed from the *D*-dimensional distances onto the final two-dimensional representation. Gower and Ross [62] show how this technique can highlight distortions in the two-dimensional representation. The Minimal Spanning Tree (MST) of a data set is a graph connecting all the data points using the shortest inter-point distances. This graph gives a kind of spine based picture of the data set structure. The idea of drawing the MST of a data set on its 2D plot has already been applied and implemented, eg. in the Spinne program [13].

Clustering data before visualization

When large data sets are visualized, it is often necessary to map first a subset and in a second step add somehow the rest of the data set. The points that constitute the subset can be taken to be the cluster centers obtained by a clustering method as eg. the k–means. The data set's MST can also be used to select a subset. Choosing the N_b highest (ie. closest to the root) vertices of the MST will lead to a basis subset that best depicts the spine of the data structure in terms of distances. The remaining data can be added by relative mapping.

Visualization that takes into account the class information

If we desire a display that separates necessarily classes, it may be necessary to incorporate the class information into the mapping process. Many strategies are possible, as eg. the one presented in the NeuroScale algorithm (see §3.2.4, page 14). A simple adaptation of the MDS algorithm that includes class labels is inspired by Fisher's discriminants and consists in minimizing the following measure:

$$M = \frac{M_i}{M_b} = \frac{\sum_i (X_i - \bar{X})(X_i - \bar{X})}{\sum_{i<j} (X_i - X)(X_j - X)} \tag{7.1}$$

where M_i is a measure of the within class variance for all the classes and M_b is a measure of the between class variances, and with $D_{ij} = \sum_k w_k \| X_i^k - X_j^k \|^2$.

Possible improvements to the software

- Multidimensional scaling, as well as self-organizing maps and principal components analysis, allow multivariate data visualization by dimensionality reduction. The distortions induced by this compression would be smaller when reducing data to 3 dimensions instead of 2 dimensions. A powerful extension to our software would be the possibility to visualize 3-dimensional data by allowing its interactive rotation. This extension concerns only the part of the software managing the display of points, because the MDS mapping procedures are already able to map data from D-space to any D'-space, with $D' \in]2, D]$.

- When zooming in a chosen area of the data space, it happens that the zoomed configuration does not match very well the display of the points before zooming (due to rotation or reflection). A first idea to avoid this defect is to initialize the mapping of the zoomed points with their configuration in the display before zooming. A Procrustes analysis (see §4.2.5) would allow to show the zoomed configuration properly rotated and reflected so that its points correspond as well as possible to the points from the configuration before zooming.

- The possibility to interactively move (using the "drag and drop" mouse technique) a mapped point with simultaneous visualization of the Stress value would be helpful in situations where one point is evidently surrounded by points from a different class and seems to be misplaced. This situation may suggest that this point was badly initialized and is trapped in a local minimum. It may occur that dragging this point, while looking at the Stress value, would enable the user to place it in an area where it seems to belong to. This would lead to an interactive point-wise Stress minimization driven by the both the display and the Stress value.

Appendix A

Optimized step-size for steepest descent minimization

The purpose of the following expressions is to simplify analytically as much as possible the expression of the optimized step-size $\alpha^{(k)}$ in order to reduce its calculation complexity.

A.1 Unified expressions for `Stress`

The unified Stress expressions are introduced in order to have the same calculation framework in the different cases of standard mapping and relative mapping. This avoids to have different calculations (and implementations) for each case. We recall that N_t is the total number of mapped points, N_m is the number of moving points, and we denote $N_f = N_t - N_m$ as the number of points (if any) that are fixed during the mapping process. The general Stress expression of equation (4.3) for standard mapping is:

$$S(\mathbf{Y}) = \sum_{i<j}^{N_t} w_{ij} \cdot \left(\delta_{ij} - d_{ij}(\mathbf{Y})\right)^2 \tag{A.1}$$

where the normalization factor F_n of equation (4.3) is included in the weights $\{w_{ij}\}$ assigned to the distances $\{d_{ij}\}$. We assume here that these weights are independent from $\mathbf{Y}$, which simplifies the subsequent derivatives expressions. In the general case of *relative mapping*, the Stress expression has been redefined in equation 4.26 as:

$$S_r(\mathbf{Y}) = \sum_{i<j}^{N_m} w_{ij} \cdot \left(\delta_{ij} - d_{ij}(\mathbf{Y})\right)^2 + \sum_{i=1}^{N_m} \sum_{j=N_m+1}^{N_t} w_{ij} \cdot \left(\delta_{ij} - d_{ij}(\mathbf{Y})\right)^2 \tag{A.2}$$

In order to simplify further calculations on Stress derivatives, we rewrite Stress expressions (A.1) and (A.2) in the following unique expression using only one index m running over all the changing distances:

$$S(\mathbf{Y}) = \sum_{m}^{N_d} w_{ab} \cdot (\delta_{ab} - d_{ab}(\mathbf{Y}))^2 \tag{A.3}$$

where N_d is the number of changing distances, that is $N_d = N_t(N_t - 1)/2$ in the case of expression (A.1) and $N_d = N_m(N_m - 1)/2 + N_m(N_t - N_m)$ in the case of expression (A.2), and (a, b) are indices of the points separated by the m^{th} distance, with $a < b$. This transfer of indices is achieved by building two correspondence index vectors $a[m]$ and $b[m]$.

A.1.1 Interpoint distances derivatives

The Euclidean distance d_{ab} separating points with indices a and b is expressed as:

$$d_{ab}(\mathbf{Y}) = \left(\sum_{z=1}^{d_{out}} (y_{az} - y_{bz})^2 \right)^{\frac{1}{2}} \tag{A.4}$$

$$\frac{\partial d_{ab}(\mathbf{Y})}{\partial y_{kl}} = \frac{1}{2d_{ab}} \cdot \frac{\partial}{\partial y_{kl}} \left(\sum_{z=1}^{d_{out}} (y_{az} - y_{bz})^2 \right) \tag{A.5}$$

$$\frac{\partial d_{ab}(\mathbf{Y})}{\partial y_{kl}} = \frac{1}{2d_{ab}} \sum_{z=1}^{d_{out}} \frac{\partial}{\partial y_{kl}} (y_{az} - y_{bz})^2 \tag{A.6}$$

$$\frac{\partial d_{ab}(\mathbf{Y})}{\partial y_{kl}} = \frac{1}{2d_{ab}} \sum_{z=1}^{d_{out}} 2\,(y_{az} - y_{bz}) \frac{\partial}{\partial y_{kl}} (y_{az} - y_{bz}) \tag{A.7}$$

Using Kronecker's symbol $\boldsymbol{\delta}^{ij}$[1], we note $\frac{\partial}{\partial y_{kl}} (y_{az} - y_{bz}) = \boldsymbol{\delta}^{lz} \left(\boldsymbol{\delta}^{ka} - \boldsymbol{\delta}^{kb} \right)$ hence:

$$\frac{\partial d_{ab}(\mathbf{Y})}{\partial y_{kl}} = \frac{1}{d_{ab}} \sum_{z=1}^{d_{out}} (y_{az} - y_{bz}) \boldsymbol{\delta}^{lz} \left(\boldsymbol{\delta}^{ka} - \boldsymbol{\delta}^{kb} \right) \tag{A.8}$$

Noting that $\boldsymbol{\delta}^{lz} \neq 0$ only when $z = l$, we get:

$$\frac{\partial d_{ab}(\mathbf{Y})}{\partial y_{kl}} = \frac{1}{d_{ab}} (y_{al} - y_{bl}) \left(\boldsymbol{\delta}^{ka} - \boldsymbol{\delta}^{kb} \right) \tag{A.9}$$

A.1.2 Stress gradient expressions

The gradient vector ∇S of Stress function S is defined as:

$$\nabla S(\mathbf{Y}) = \left(\frac{\partial S(\mathbf{Y})}{\partial y_{kl}} \right)_{k=1,\dots,N_m,l=1,\dots,d_{out}} \tag{A.10}$$

Using Stress expression of equation (A.3):

$$\frac{\partial S(\mathbf{Y})}{\partial y_{kl}} = \frac{\partial}{\partial y_{kl}} \left(\frac{1}{F_n} \sum_{m=1}^{N_d} w_{ab} \cdot (\delta_{ab} - d_{ab})^2 \right) \tag{A.11}$$

Assuming that F_n and w_{ab} are independent from $\mathbf{Y}$, we get:

$$\frac{\partial S(\mathbf{Y})}{\partial y_{kl}} = \frac{1}{F_n} \sum_{m=1}^{N_d} w_{ab} \cdot 2\,(\delta_{ab} - d_{ab}) \left(-\frac{\partial d_{ab}}{\partial y_{kl}} \right) \tag{A.12}$$

Using expression for $\frac{\partial d_{ab}}{\partial y_{kl}}$ of equation (A.9), we get:

$$\frac{\partial S(\mathbf{Y})}{\partial y_{kl}} = \frac{-2}{F_n} \sum_{m=1}^{N_d} w_{ab} \left(\boldsymbol{\delta}^{ka} - \boldsymbol{\delta}^{kb} \right) \left(\frac{\delta_{ab} - d_{ab}}{d_{ab}} \right) (y_{al} - y_{bl}) \tag{A.13}$$

[1] Kronecker's symbol $\boldsymbol{\delta}^{ij}$ should not be confused with the dissimilarity symbol δ_{ij}.

A.1.3 Stress Hessian matrix expressions

The Hessian matrix is the matrix of second derivatives defined as:

$$H_S(\mathbf{Y}) = \left[\frac{\partial^2 S(\mathbf{Y})}{\partial y_{ij} \partial y_{kl}}\right]_{i=1,\dots,N_m, k=1,\dots,N_m; j=1\dots,d_{out}, l=1,\dots,d_{out}} \tag{A.14}$$

Using Stress gradient expression (A.13):

$$\frac{\partial^2 S(\mathbf{Y})}{\partial y_{ij} \partial y_{kl}} = \frac{\partial}{\partial y_{ij}}\left(\frac{\partial S}{\partial y_{kl}}\right) = \frac{\partial}{\partial y_{ij}}\left(\frac{-2}{F_n}\sum_{m=1}^{N_d} w_{ab}\left(\boldsymbol{\delta}^{ka} - \boldsymbol{\delta}^{kb}\right)\left(\frac{\delta_{ab} - d_{ab}}{d_{ab}}\right)(y_{al} - y_{bl})\right) \tag{A.15}$$

Assuming that F_n and w_{ab} are independent from $\mathbf{Y}$, we get:

$$\frac{\partial^2 S(\mathbf{Y})}{\partial y_{ij} \partial y_{kl}} = \frac{-2}{F_n}\sum_{m=1}^{N_d} w_{ab}\left(\boldsymbol{\delta}^{ka} - \boldsymbol{\delta}^{kb}\right)\frac{\partial}{\partial y_{ij}}\left(\left(\frac{\delta_{ab} - d_{ab}}{d_{ab}}\right)(y_{al} - y_{bl})\right) \tag{A.16}$$

$$\frac{\partial^2 S(\mathbf{Y})}{\partial y_{ij} \partial y_{kl}} = \frac{-2}{F_n}\sum_{m=1}^{N_d} w_{ab}\left(\boldsymbol{\delta}^{ka} - \boldsymbol{\delta}^{kb}\right)\left(\ \tfrac{\partial}{\partial y_{ij}}\left(\tfrac{\delta_{ab}-d_{ab}}{d_{ab}}\right)(y_{al} - y_{bl}) + \left(\tfrac{\delta_{ab}-d_{ab}}{d_{ab}}\right)\tfrac{\partial}{\partial y_{ij}}(y_{al} - y_{bl}) \ \right) \tag{A.17}$$

$$\frac{\partial^2 S(\mathbf{Y})}{\partial y_{ij} \partial y_{kl}} = \sum_{m=1}^{N_d} w_{ab}\left(\boldsymbol{\delta}^{ka} - \boldsymbol{\delta}^{kb}\right)\left(\begin{array}{c} -\tfrac{\delta_{ab}}{d_{ab}^2}\tfrac{\partial d_{ab}}{\partial y_{ij}}(y_{al} - y_{bl}) \\ +\left(\tfrac{\delta_{ab}-d_{ab}}{d_{ab}}\right)\tfrac{\partial}{\partial y_{ij}}(y_{al} - y_{bl}) \end{array}\right) \tag{A.18}$$

Using expression for $\frac{\partial d_{ab}}{\partial y_{kl}}$ of (A.9) and noting $\frac{\partial}{\partial y_{ij}}(y_{al} - y_{bl}) = \boldsymbol{\delta}^{jl}\left(\boldsymbol{\delta}^{ia} - \boldsymbol{\delta}^{ib}\right)$, we get:

$$\frac{\partial^2 S(\mathbf{Y})}{\partial y_{ij} \partial y_{kl}} = \frac{-2}{F_n}\sum_{m=1}^{N_d} w_{ab}\left(\boldsymbol{\delta}^{ka} - \boldsymbol{\delta}^{kb}\right)\left(\begin{array}{c} -\tfrac{\delta_{ab}}{d_{ab}^2}\tfrac{1}{d_{ab}}(y_{aj} - y_{bj})\left(\boldsymbol{\delta}^{ia} - \boldsymbol{\delta}^{ib}\right)\cdot(y_{al} - y_{bl}) \\ +\left(\tfrac{\delta_{ab}-d_{ab}}{d_{ab}}\right)\boldsymbol{\delta}^{jl}\left(\boldsymbol{\delta}^{ia} - \boldsymbol{\delta}^{ib}\right) \end{array}\right) \tag{A.19}$$

$$\frac{\partial^2 S(\mathbf{Y})}{\partial y_{ij} \partial y_{kl}} = \frac{-2}{F_n}\sum_{m=1}^{N_d} w_{ab}\left(\boldsymbol{\delta}^{ia} - \boldsymbol{\delta}^{ib}\right)\left(\boldsymbol{\delta}^{ka} - \boldsymbol{\delta}^{kb}\right)\left(\begin{array}{c} -\tfrac{\delta_{ab}}{d_{ab}^3}(y_{aj} - y_{bj})(y_{al} - y_{bl}) \\ +\boldsymbol{\delta}^{jl}\left(\tfrac{\delta_{ab}-d_{ab}}{d_{ab}}\right) \end{array}\right) \tag{A.20}$$

In a purpose of notation simplification, we define $\Delta_{abik} = \left(\boldsymbol{\delta}^{ia} - \boldsymbol{\delta}^{ib}\right)\left(\boldsymbol{\delta}^{ka} - \boldsymbol{\delta}^{kb}\right)$, hence:

$$\frac{\partial^2 S(\mathbf{Y})}{\partial y_{ij} \partial y_{kl}} = \frac{-2}{F_n}\sum_{m=1}^{N_d} w_{ab}\Delta_{abik}\left(-\frac{\delta_{ab}}{d_{ab}^3}(y_{aj} - y_{bj})(y_{al} - y_{bl}) + \boldsymbol{\delta}^{jl}\left(\frac{\delta_{ab} - d_{ab}}{d_{ab}}\right)\right) \tag{A.21}$$

A.1.4 Optimal step-size expressions

The optimal step-size at iteration k has been expressed in (4.24) as:

$$\alpha^{(k)} = \frac{\nabla S^{(k)^T} \cdot \nabla S^{(k)}}{\nabla S^{(k)^T} \cdot H_S^{(k)} \cdot \nabla S^{(k)}} = \frac{\|\nabla S^{(k)}\|^2}{\|\nabla S^{(k)}\|_H^2} \tag{A.22}$$

Let us define $\alpha_n = \nabla S^{(k)^T} \cdot \nabla S^{(k)}$ and $\alpha_d = \nabla S^{(k)^T} \cdot H_S^{(k)} \cdot \nabla S^{(k)}$, so that $\alpha^{(k)} = \alpha_n / \alpha_d$. Using sum notations we have:

$$\alpha_n = \sum_{k=1}^{N_m} \sum_{l=1}^{d_{out}} \left(\frac{\partial S}{\partial y_{kl}} \right)^2 \tag{A.23}$$

At each iteration k, all the components $\left(\frac{\partial S}{\partial x_k^l} \right), k = 1, ..., N_m, l = 1, ..., d_{out}$ of the gradient $\nabla S^{(k)}$ must be calculated in order to get the steepest descent direction, so expression (A.23) of α_n does not need to be further simplified.

$$\alpha_d = \sum_{i=1}^{N_m} \sum_{j=1}^{d_{out}} \left(\frac{\partial S}{\partial y_{ij}} \sum_{k=1}^{N_m} \sum_{l=1}^{d_{out}} \left(\frac{\partial^2 S}{\partial y_{ij} \partial y_{kl}} \cdot \frac{\partial S}{\partial y_{kl}} \right) \right) \tag{A.24}$$

The symmetry of this expression and of the Hessian matrix H_S will lead to analytic simplifications for α_d. Using second derivatives expressions of equation (A.21):

$$\alpha_d = \sum_{i=1}^{N_m} \sum_{j=1}^{d_{out}} \left(\frac{\partial S}{\partial y_{ij}} \sum_{k=1}^{N_m} \sum_{l=1}^{d_{out}} \frac{-2}{F_n} \sum_{m=1}^{N_d} w_{ab} \Delta_{abik} \begin{pmatrix} -\frac{\delta_{ab}}{d_{ab}^3} \left(y_{aj} - y_{bj} \right) \left(y_{al} - y_{bl} \right) \\ +\boldsymbol{\delta}^{jl} \left(\frac{\delta_{ab} - d_{ab}}{d_{ab}} \right) \end{pmatrix} \cdot \frac{\partial S}{\partial y_{kl}} \right) \tag{A.25}$$

We can reorder the sums as follows:

$$\alpha_d = \frac{-2}{F_n} \sum_{m=1}^{N_d} w_{ab} \sum_{j=1}^{d_{out}} \sum_{i=1}^{N_m} \left(\frac{\partial S}{\partial y_{ij}} \sum_{k=1}^{N_m} \Delta_{abik} \sum_{l=1}^{d_{out}} \begin{pmatrix} -\frac{\delta_{ab}}{d_{ab}^3} \left(y_{aj} - y_{bj} \right) \left(y_{al} - y_{bl} \right) \\ +\boldsymbol{\delta}^{jl} \left(\frac{\delta_{ab} - d_{ab}}{d_{ab}} \right) \end{pmatrix} \cdot \frac{\partial S}{\partial y_{kl}} \right) \tag{A.26}$$

Splitting the sum over l into two parts:

$$\alpha_d = \frac{-2}{F_n} \sum_{m=1}^{N_d} w_{ab} \sum_{j=1}^{d_{out}} \sum_{i=1}^{N_m} \left(\frac{\partial S}{\partial y_{ij}} \sum_{k=1}^{N_m} \Delta_{abik} \begin{pmatrix} -\sum_{l=1}^{d_{out}} \frac{\delta_{ab}}{d_{ab}^3} \left(y_{aj} - y_{bj} \right) \left(y_{al} - y_{bl} \right) \frac{\partial S}{\partial y_{kl}} \\ +\sum_{l=1}^{d_{out}} \boldsymbol{\delta}^{jl} \left(\frac{\delta_{ab} - d_{ab}}{d_{ab}} \right) \frac{\partial S}{\partial y_{kl}} \end{pmatrix} \right) \tag{A.27}$$

Noting that $\boldsymbol{\delta}^{jl} \neq 0$ only when $j = l$, we get:

$$\alpha_d = \frac{-2}{F_n} \sum_{m=1}^{N_d} w_{ab} \sum_{j=1}^{d_{out}} \sum_{i=1}^{N_m} \left(\frac{\partial S}{\partial y_{ij}} \sum_{k=1}^{N_m} \Delta_{abik} \begin{pmatrix} -\sum_{l=1}^{d_{out}} \frac{\delta_{ab}}{d_{ab}^3} \left(y_{aj} - y_{bj} \right) \left(y_{al} - y_{bl} \right) \frac{\partial S}{\partial y_{kl}} \\ +\left(\frac{\delta_{ab} - d_{ab}}{d_{ab}} \right) \frac{\partial S}{\partial y_k^j} \end{pmatrix} \right) \tag{A.28}$$

Knowing that $a \neq b$, and that $\Delta_{abik} = 0$ unless $(i = a \vee i = b) \wedge (k = a \vee k = b)$, we get:

$$\alpha_d = \frac{-2}{F_n} \sum_{m=1}^{N_d} w_{ab} \sum_{j=1}^{d_{out}} \begin{pmatrix} \frac{\partial S}{\partial y_{aj}} \begin{pmatrix} -\sum_{l=1}^{d_{out}} \frac{\delta_{ab}}{d_{ab}^3} \left(y_{aj} - y_{bj} \right) \left(y_{al} - y_{bl} \right) \frac{\partial S}{\partial y_{al}} + \left(\frac{\delta_{ab} - d_{ab}}{d_{ab}} \right) \frac{\partial S}{\partial y_{aj}} \\ +\sum_{l=1}^{d_{out}} \frac{\delta_{ab}}{d_{ab}^3} \left(y_{aj} - y_{bj} \right) \left(y_{al} - y_{bl} \right) \frac{\partial S}{\partial y_{bl}} - \left(\frac{\delta_{ab} - d_{ab}}{d_{ab}} \right) \frac{\partial S}{\partial y_{bj}} \end{pmatrix} \\ +\frac{\partial S}{\partial y_{bj}} \begin{pmatrix} +\sum_{l=1}^{d_{out}} \frac{\delta_{ab}}{d_{ab}^3} \left(y_{aj} - y_{bj} \right) \left(y_{al} - y_{bl} \right) \frac{\partial S}{\partial y_{al}} - \left(\frac{\delta_{ab} - d_{ab}}{d_{ab}} \right) \frac{\partial S}{\partial y_{aj}} \\ -\sum_{l=1}^{d_{out}} \frac{\delta_{ab}}{d_{ab}^3} \left(y_{aj} - y_{bj} \right) \left(y_{al} - y_{bl} \right) \frac{\partial S}{\partial y_{bl}} + \left(\frac{\delta_{ab} - d_{ab}}{d_{ab}} \right) \frac{\partial S}{\partial y_{bj}} \end{pmatrix} \end{pmatrix} \tag{A.29}$$

$$\alpha_d = \frac{-2}{F_n}\sum_{m=1}^{N_d} w_{ab}\sum_{j=1}^{d_{out}}\left(\left(\frac{\partial S}{\partial y_{aj}}-\frac{\partial S}{\partial y_{bj}}\right)\left(\begin{array}{c} -\sum_{l=1}^{d_{out}}\frac{\delta_{ab}}{d_{ab}^3}\left(y_{aj}-y_{bj}\right)\left(y_{al}-y_{bl}\right)\frac{\partial S}{\partial y_{al}}+\left(\frac{\delta_{ab}-d_{ab}}{d_{ab}}\right)\frac{\partial S}{\partial y_{aj}} \\ +\sum_{l=1}^{d_{out}}\frac{\delta_{ab}}{d_{ab}^3}\left(y_{aj}-y_{bj}\right)\left(y_{al}-y_{bl}\right)\frac{\partial S}{\partial y_{bl}}-\left(\frac{\delta_{ab}-d_{ab}}{d_{ab}}\right)\frac{\partial S}{\partial y_{bj}} \end{array}\right)\right) \tag{A.30}$$

$$\alpha_d = \frac{-2}{F_n}\sum_{m=1}^{N_d} w_{ab}\sum_{j=1}^{d_{out}}\left(\left(\frac{\partial S}{\partial y_{aj}}-\frac{\partial S}{\partial y_{bj}}\right)\left(\begin{array}{c} -\frac{\delta_{ab}}{d_{ab}^3}\left(y_{aj}-y_{bj}\right)\sum_{l=1}^{d_{out}}\left(y_{al}-y_{bl}\right)\left(\frac{\partial S}{\partial y_{al}}-\frac{\partial S}{\partial y_{bl}}\right) \\ +\left(\frac{\delta_{ab}-d_{ab}}{d_{ab}}\right)\left(\frac{\partial S}{\partial y_{aj}}-\frac{\partial S}{\partial y_{bj}}\right) \end{array}\right)\right) \tag{A.31}$$

A.2 Expressions for `SStress`

Using the same calculus scheme as in section (A.1), similar expressions have been derived for the `SStress` function, defined in §4.4.1 as

$$SS(\mathbf{Y}) = \frac{1}{F_n}\sum_{i<j}^{N_t} w_{ij}\cdot\left(\delta_{ij}^2-d_{ij}^2(\mathbf{Y})\right)^2 \tag{A.32}$$

First, the squared distance derivatives

$$\frac{\partial d_{ab}^2(\mathbf{Y})}{\partial y_{kl}} = 2\left(y_{al}-y_{bl}\right)\left(\boldsymbol{\delta}^{ka}-\boldsymbol{\delta}^{kb}\right), \tag{A.33}$$

then the `SStress` gradient vector

$$\frac{\partial SS(\mathbf{Y})}{\partial y_{kl}} = \frac{-4}{F_n}\sum_{m=1}^{N_d} w_{ab}\left(\boldsymbol{\delta}^{ka}-\boldsymbol{\delta}^{kb}\right)\left(\delta_{ab}^2-d_{ab}^2\right)\left(y_{al}-y_{bl}\right) \tag{A.34}$$

and the corresponding Hessian matrix

$$\frac{\partial^2 SS(\mathbf{Y})}{\partial y_{ij}\partial y_{kl}} = \frac{-4}{F_n}\sum_{m=1}^{N_d} w_{ab}\Delta_{abik}\left(-2\left(y_{aj}-y_{bj}\right)\left(y_{al}-y_{bl}\right)+\boldsymbol{\delta}^{jl}\left(\delta_{ab}^2-d_{ab}^2\right)\right) \tag{A.35}$$

leading to the optimized step-size's denominator

$$\alpha_d = \frac{-4}{F_n}\sum_{m=1}^{N_d} w_{ab}\sum_{j=1}^{d_{out}}\left(\left(\frac{\partial S}{\partial y_{aj}}-\frac{\partial S}{\partial y_{bj}}\right)\left(\begin{array}{c} -2\left(y_{aj}-y_{bj}\right)\sum_{l=1}^{d_{out}}\left(y_{al}-y_{bl}\right)\left(\frac{\partial S}{\partial y_{al}}-\frac{\partial S}{\partial y_{bl}}\right) \\ +\left(\delta_{ab}^2-d_{ab}^2\right)\left(\frac{\partial S}{\partial y_{aj}}-\frac{\partial S}{\partial y_{bj}}\right) \end{array}\right)\right) \tag{A.36}$$

Appendix B

Outline of the interactive software

In this appendix, the functionalities of the MDS based interactive software that we developed as a tool for multivariate data visualization are presented. This interactive software is an MDI (Multiple Document Interface) type software, which means that several documents (in this case data sets) can be open and processed simultaneously. The computations related to the mapping of one data set are run in a separate thread. The various commands available in this software can be grouped into two categories: Commands applying directly to data sets and commands related to the mapping operation. This allows us to define two basic classes of objects: the `DataSet` class and the `Mapping` class.

B.1 The `DataSet` class

`Dataset` menu: Data set basic operations

- **Open** Opens a disk file containing multivariate data in the specific data format,

- **Pre-processing** Various data set pre-processings are available:

  - **Single points** Removes redundancies from the data to keep only single data points,
  - **Normalize** Performs normalization (0,1) of the current data set, so that in each dimension the midrange is 0 and the range is 1. Let $\tilde{x}_{ij}$ be the normalized value of x_{ij} :

$$\tilde{x}_{ij} = \frac{x_{ij} - \mathit{midrange}\ x_j}{\mathit{range}\ x_j} \tag{B.1a}$$

where:

$$\mathit{midrange}\ x_j = \frac{\max_i x_{ij} + \min_i x_{ij}}{2} \tag{B.1b}$$

$$\mathit{range}\ x_j = \max_i x_{ij} - \min_i x_{ij} \tag{B.1c}$$

  - **Standardize** Performs standardization (0,1) of the current data set, so that in each dimension the mean is 0 and standard deviation is 1. Let $\tilde{x}_{ij}$ be the standardized value of x_{ij} :

$$\tilde{x}_{ij} = \frac{x_{ij} - \bar{x}_j}{s_j} \tag{B.2a}$$

where:

$$\bar{x}_j = \frac{1}{N}\sum_{i=1}^{N} x_{ij} \tag{B.2b}$$

$$s_j^2 = \frac{1}{N-1}\sum_{i=1}^{N}(x_{ij} - \bar{x}_j)^2 \tag{B.2c}$$

  - **Scale features** Re-scale, that is multiply by some given factors, some features of the current data set. This allows to increase or decrease the weight of chosen dimensions.[1]

- **Save** Saves the current Data set in a file. A dialog box allows to choose which data points to save, which features and in which format (the specific data format or in a PostScript file).

- **Print** Sends the bitmap of the active `DataSet`'s Table View to the printer,

- **Close** Closes the current Data set. There is no automatic saving. If the data set is a new one (e.g. a mapping result that has not been saved yet), it should be saved prior to closing using the **Save** option.

Data format

Data file are editable ASCII text files, their names should have a '`.dat`' extension. The first line contains the data dimension *D*, each following line describes one data point in the feature space: the *D* features followed by the class label. Lines beginning with # are ignored and can contain comments. Missing data values must be marked as `?`

```
<data_dimension>
<first_feature> ...  <last_feature> <class_label> (← First data)
...
<first_feature> ...  <last_feature> <class_label> (← Last data)
```

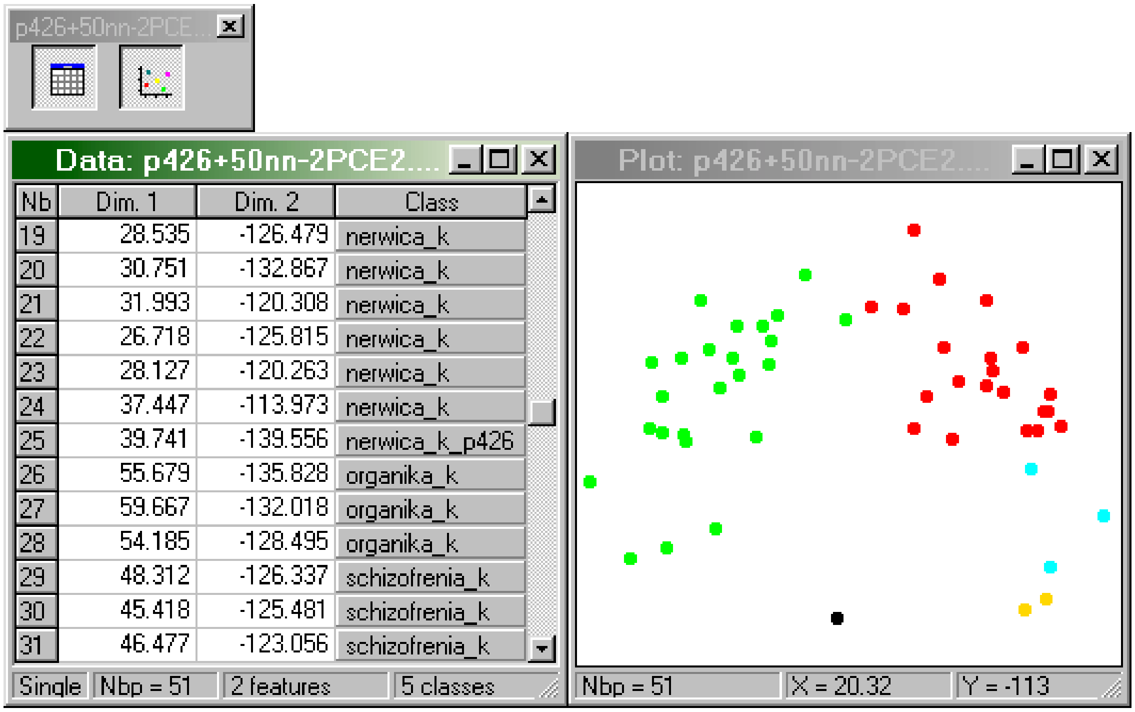
A 'new line' character must terminate the last data line.

`DataSet` object

A small window represents this object with the data set name in the title bar, and containing two buttons. The multivariate data can be viewed in a table (left button) called Data View in which each row is a data point (or sample or case). The first column (header `Nb`) contains an index number. The following columns are dimensions (or attributes or features), and the last column contains the class name to which this data belongs. The data can be viewed in two ways or modes: the `Single` mode in which only single data points are displayed, and the `Origin` mode where all the data are shown as in the original data set. This functionality is necessary because MDS algorithm can't be used on data with identical cases (see §6.3.2). If two data points have exactly the same features, one of them has to be removed or moved (that is modified by adding a small random noise) before mapping. The current view mode is

[1] This is helpful when visualizing a classifier's behavior on a given data set, for which some dimensions must be weighted to reach better classification results.

displayed on the bottom Status Bar, on the left side. The Status Bar also shows the number Nbp of displayed data (single or original, according to current mode). A right mouse Popup Menu allows to switch between those two modes. After mapping, the data are saved according to their original number. Data points selection can be done directly on the Table View by clicking the point header (index). Data dimensions selection can be done directly on the Table View by a left click on the dimension header (feature name or number). Multivariate data can be viewed on a scatterplot (right button) called Plot View in which the horizontal axis (X) is initially set to the first feature and the vertical axis (Y) to the second feature.



Nb	Dim. 1	Dim. 2	Class
19	28.535	-126.479	nerwica_k
20	30.751	-132.867	nerwica_k
21	31.993	-120.308	nerwica_k
22	26.718	-125.815	nerwica_k
23	28.127	-120.263	nerwica_k
24	37.447	-113.973	nerwica_k
25	39.741	-139.556	nerwica_k_p426
26	55.679	-135.828	organika_k
27	59.667	-132.018	organika_k
28	54.185	-128.495	organika_k
29	48.312	-126.337	schizofrenia_k
30	45.418	-125.481	schizofrenia_k
31	46.477	-123.056	schizofrenia_k

Single | Nbp = 51 | 2 features | 5 classes

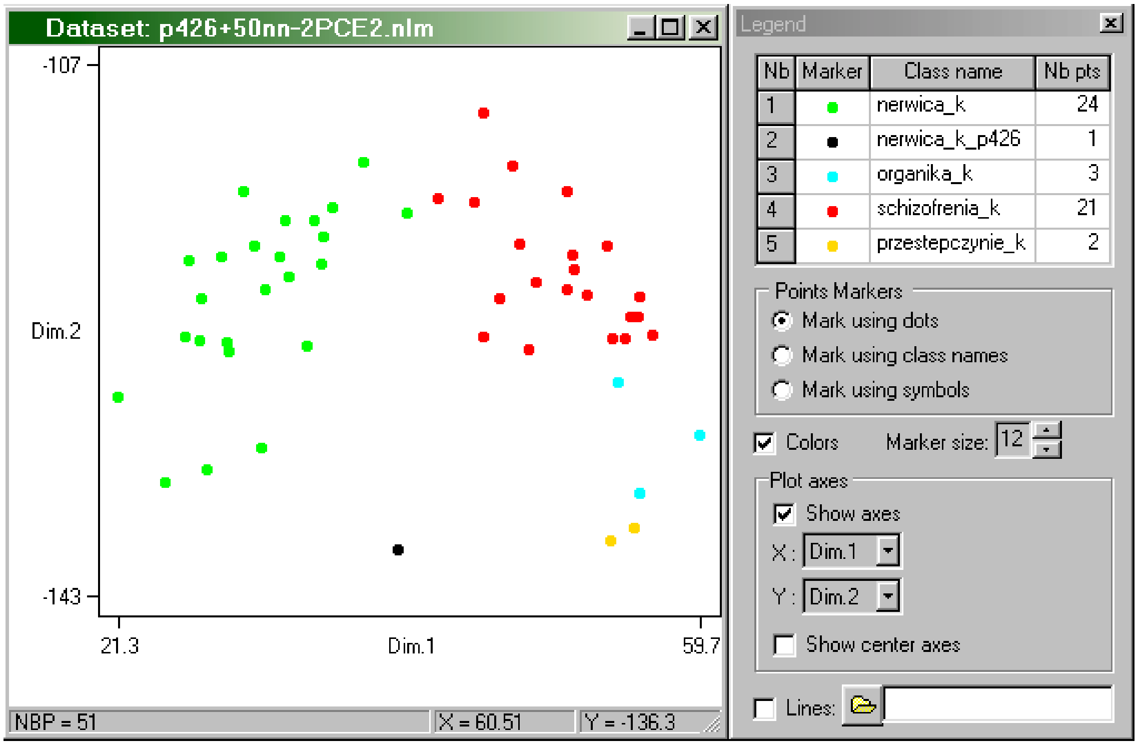
Nbp = 51 | X = 20.32 | Y = -113

Figure B.1: The IMDS software: a data set with its data and plot views.

The choice of which features are represented on the plot view and other parameters controlling the aspect of the plot are made in the Legend dialog box.

The Legend dialog box

The Legend dialog box contains on its top a table describing the classes of the active DataSet. Each line is for one class, with an index number (header Nb), the current class marker, the class name and number of points. Three types of point's marker are available in the Points Markers Radio Group: a dot (default), the class name or a geometric symbol. Those markers can be colored or black, as set in the Colors Check Box. The marker's size can be set using the Marker size Up-Down Button. The user can change the marker's color or symbol for one or more class (by a right mouse click on the marker) to another value from a set of 32 given possibilities. The plot axes are displayed if the user checks the Show axes Check Box. The features or dimensions that are represented by the X and Y coordinates are set in the two Combo Boxes labeled X and Y. If the Show center axes Check Box is checked, two axes are displayed in the mean range of the minima and maxima values on both axis.



Dataset: p426+50nn-2PCE2.nlm

NBP = 51 | X = 60.51 | Y = -136.3

Legend

Nb	Marker	Class name	Nb pts
1	🟢	nerwica_k	24
2	⚫	nerwica_k_p426	1
3	🔵	organika_k	3
4	🔴	schizofrenia_k	21
5	🟡	przestepczynie_k	2

Points Markers
● Mark using dots
○ Mark using class names
○ Mark using symbols

☑ Colors   Marker size: 12

Plot axes
☑ Show axes
X : Dim.1
Y : Dim.2
☐ Show center axes

☐ Lines:

Figure B.2: The `IMDS` software: plot view of a data set and its `Legend` dialog box.

The `Points selection` dialog box

A possibility for selecting data points is offered to the user in the following purposes:
a) Just save (in an ASCII or PostScript file) a selected subset of an open data set,
b) Map (or map again) a subset of an open data set. This is necessary when mapping points in the neighborhood of one data point (dynamic zooming).

This dialog box allows to select or to deselect data points of the active `DataSet`. Selected points are represented on the Plot View by a red circle around the dot marker, and by standard text selection on the Table View. The total number of selected points for the current data set is displayed on the bottom Status Bars (between parentheses, next to the number of points) of its Data and Plot Views. Selection is realized by first clicking the appropriate button for the desired technique (on the left column) and then clicking if necessary on the Plot or Data View. Similarly, clicking the appropriate button in the right column allows to deselect some previously selected points. The six different methods available to select or deselect data points are the following:

- **1 point**: one point is selected individually by clicking on its marker on the Plot View or on its index number (header `Nb`) in the Data View,

- **disc r**: the data points located within a disc of given radius r are selected. Set the value of r in the r = Up-Down Control on the right side of the dialog box and click in the Plot View to locate the disc center,

- **2D knn**: the k nearest neighbors of a given point p_c in the 2-space are selected. Set the value of k in the k = Up-Down Control on the right side of the dialog box and click the marker of the point p_c on the Plot View or its index number (header `Nb`) in the Data View,

- **ND knn**: the k nearest neighbors of a given point P_c in the D-space are selected. Set the value of *k* in the k = Up-Down Control on the right side of the dialog box and click the marker of the point p_c on the Plot View or its index number (header `Nb`) in the Data View,

- **rectangle**: the data points located inside a rectangle on the Plot View are selected. The rectangle is defined by left clicking when pointing its upper-left corner and dragging it down to the bottom-right corner,

- **all points**: all the data points of the current data set are selected.

Chosen dimensions can be selected as well by a left click on their header in the Data View, the selected column appears in standard text selection (click again the header of a selected dimension to deselect it). Selection of dimensions is useful when the user wants to map a data set using only a restricted number of its features (the selected dimensions), or when he wants to save only some selected features of an open data set in a new data file.

Finally, all the data points belonging to a chosen class can be selected by left clicking the chosen class index number (header `Nb`) in the top array of the `Legend` dialog box.

The `Zooming` dialog box

The `Zooming` dialog allows the user to manipulate the display of points in the Plot View of a `DataSet`. This tool is useful in the following situations: a) the user wants to compare two configurations obtained by different mapping runs by making the two configurations fit as well as possible to one another (This is a manual Procrustes analysis), b) when some areas of the viewed data set are densely filled with data points, the user is able to see them enlarged (zoomed in) and can more easily select some data points in these regions, and c) the user wants to adjust the view port window to the area interesting him and then save the view as it is in an Encapsulated PostScript file. The user is offered six different tools to manipulate the plot view, the first four are manipulation tools, the last two are more properly zooming tools. Zooming is done by first clicking the appropriate button for the desired technique and then clicking on the Plot View if necessary:

- **Initial display**: resets the view to the initial display in which the viewing area fits the data area. All the data points are visible and fit the entire view port window,

- **Pan**: translates the viewing area using the “drag and drop” technique, all the display is translated following the mouse moves,

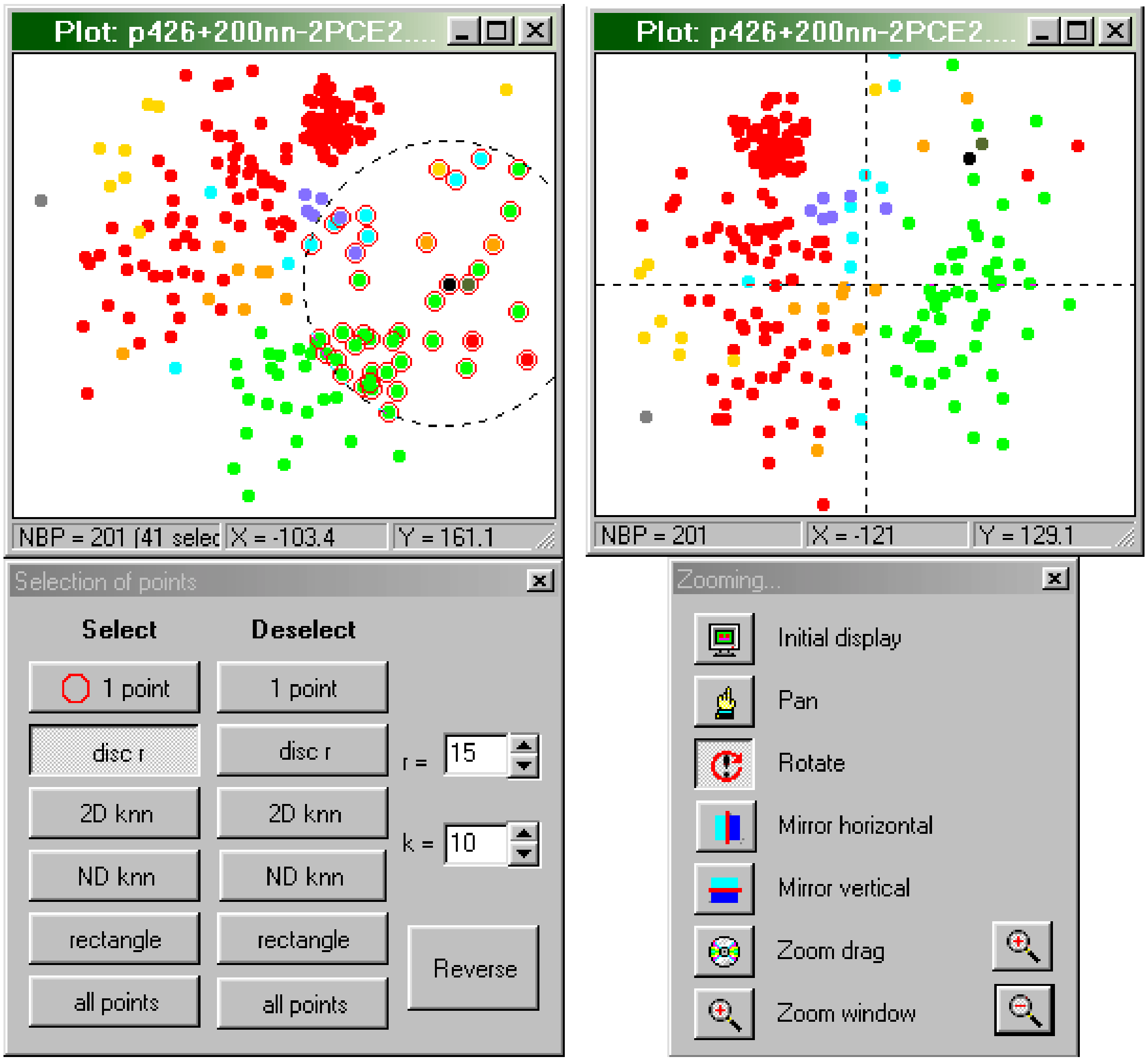
- **Rotate**: rotates the viewing area using the “drag and drop” technique, all the display is rotated around its center following the mouse moves,

- **Mirror horizontal**: reflects the configuration with respect to a vertical axis at the center of the current view, just click the button to flip (or flip back) the configuration,

- **Mirror vertical**: reflects the configuration with respect to an horizontal axis at the center of the current view, just click the button to flip (or flip back) the configuration,

- **Zoom drag**: zooms the viewing area using the “drag and drop” technique. Click on the plot and drag towards the plot center to zoom out, or towards the plot outer borders to zoom in,

- **Zoom window**: zooms into a rectangle defined using the “drag and drop” technique (as for Points Selection within a rectangle).



(a) Data `selection` dialog box.

(b) Data `zooming` dialog box.

Figure B.3: The `IMDS` software: Data `selection` and `zooming` dialog boxes.

B.2 The `Mapping` class

The visualization of a multivariate data set is obtained by performing a mapping transformation on the corresponding `DataSet` object, the result of which is the creation of a new `DataSet` object with the mapped data. This mapping action is realized by executing a function of a `Mapping` object. The information needed by this object to perform the mapping operation are of 3 kinds, they are presented in the following 3 sections.

Mapping Data

Information concerning the data sets involved with this mapping, that is input data set (data to be mapped), input fixed data set (if relative mapping is performed) and output data set (the mapped data), are given here. Additionally if some points or features are selected in the input `DataSet` object, the user can choose to map only the selected points (or all of them), taking into account only the selected features (or all of them).

Mapping Options

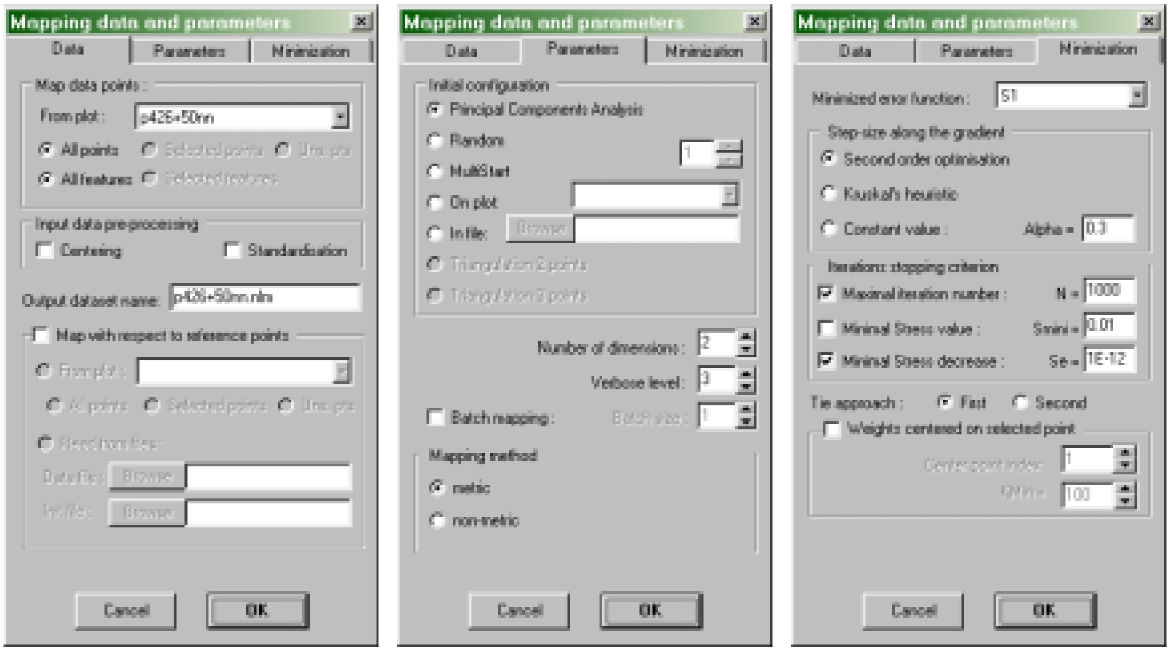
Mapping options that are independent from the employed mapping method are specified here. Those general mapping options are:

- the type of initial (or starting) configuration that can be whether: PCA (the first Principal Components), Random, Multistart, a 2-dimensional extern data set or, in the case of relative mapping with a fixed data set, by some interpolation,
- the number of dimensions of the output space (default=2),
- the type of mapping: whether a complete mapping (default) or a batch mapping, and the batch size,
- the mapping method: whether metric NLM (default) or non-metric MDS,

Mapping Parameters

The parameters connected directly and specific to the chosen mapping method (NLM or MDS) can be set here. The default choice is underlined.

- the type of the minimized Stress function (S1, S2, S3),
- the type of weights assigned to the distances in Stress function (no),
- the type of step-size along the gradient (second order optimized, Kruskal's heuristic, constant value),
- the type of stopping criterion of the iterative minimization process, that can be any combination of the following 3 criteria: maximal number of iterations, minimal Stress value reached or maximal Stress decrease,
- the tie approach (first or second, for non-metric MDS only).



Map data points :
From plot : p426+50nn
● All points ○ Selected points ○ Uns. pts
● All features ○ Selected features
Input data pre-processing
☐ Centering ☐ Standardisation
Output dataset name: p426+50nn.nlm
☐ Map with respect to reference points
○ From plot :
○ All points ○ Selected points ○ Uns. pts
○ Read from files :
Data file: Browse
Inf file: Browse

(a) `Mapping data` page.

Initial configuration
● Principal Components Analysis
○ Random 1
○ MultiStart
○ On plot
○ In file: Browse
○ Triangulation 2 points
○ Triangulation 3 points
Number of dimensions : 2
Verbose level : 3
☐ Batch mapping : Batch size : 1
Mapping method
● metric
○ non-metric

(b) `Mapping parameters` page.

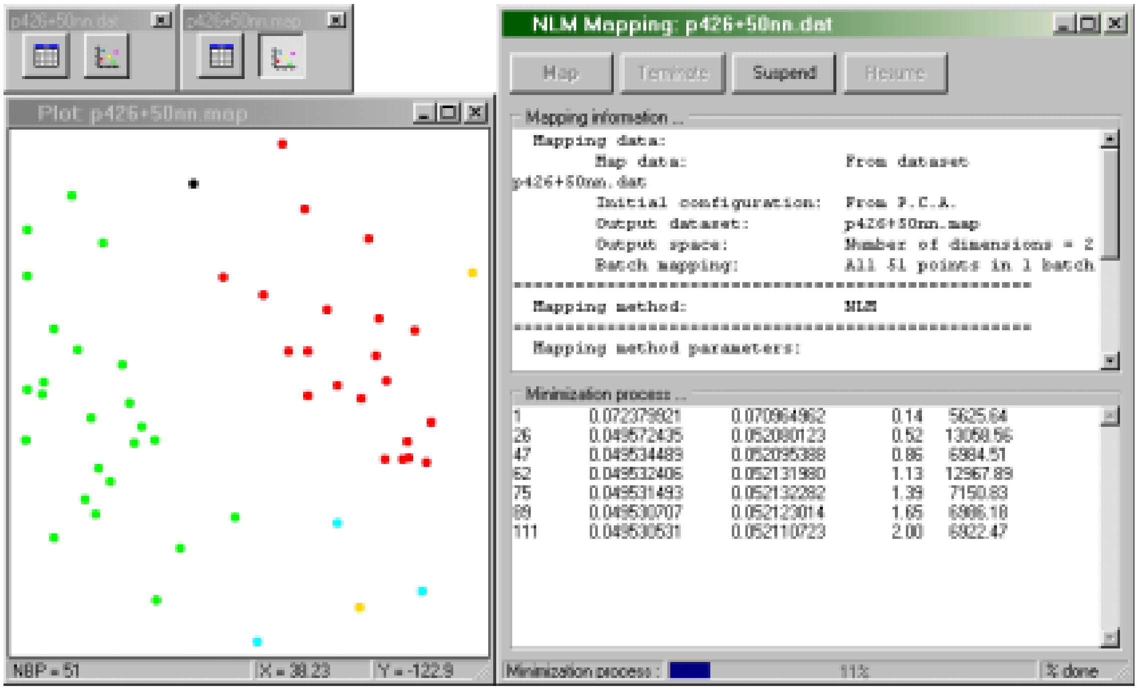
Minimized error function : S1
Step-size along the gradient
● Second order optimisation
○ Kruskal's heuristic
○ Constant value : Alpha = 0.3
Iterations stopping criterion
☑ Maximal iteration number : N = 1000
☐ Minimal Stress value : Smini = 0.01
☑ Minimal Stress decrease : Se = 1E-12
Tie approach : ● First ○ Second
☐ Weights centered on selected point
Center point index 1
Wmin = 100

(c) `Mapping method` page.

Figure B.4: The `IMDS` software: the Mapping dialog box and its three pages.

Mapping run

When the mapping data, parameters and method are set and validated (`OK` click in the Mapping data and parameters window), we get to the `Mapping run` window, from which we can start the mapping process. A new data set object is automatically created that contains the mapped data. If the plot view of this data set is shown, the user will see the evolution of the configuration during the mapping. The mapping process is ran in a separate thread that has the charge of the calculations. This thread is started by a click on the `Start` button, suspended by a click on the `suspend` button, resumed by a click on the `resume` button and terminated by a click on the `terminate` button. This gives the user the entire control on the mapping process. General information about the mapping (mapping data and parameters) is contained in the `Mapping information` Text Memo. The evolution of various values (the current iteration number, the current Stress value, the current absolute Stress value, the elapsed time in seconds and current step-size) connected to the minimization process is displayed and updated on-line in the `Minimization process` Text Memo.



```
  Mapping data:
        Map data:                   From dataset
p426+50nn.dat
        Initial configuration:      From P.C.A.
        Output dataset:             p426+50nn.map
        Output space:               Number of dimensions = 2
        Batch mapping:              All 51 points in 1 batch
=================================================
  Mapping method:                   NLM
=================================================
  Mapping method parameters:
```

```
1       0.072379821     0.070964962     0.14    5625.64
26      0.049572435     0.052080123     0.52    13058.56
47      0.049534489     0.052095388     0.86    6984.51
62      0.049532406     0.052131980     1.13    12967.89
75      0.049531493     0.052132282     1.39    7150.83
89      0.049530707     0.052123014     1.65    6906.18
111     0.049530531     0.052110723     2.00    6922.47
```

Figure B.5: The IMDS software: Mapping run window and plot view of the mapped data set.

When the mapping is terminated, the user can close the Mapping run window and possibly save the mapped data set in a file by clicking the Save button.

Bibliography

[1] Esa Alhoniemi, Jaakko Hollmén, Olli Simula, and Juha Vesanto. Process monitoring and modeling using the Self-Organizing Map. *Integrated Computer Aided Engineering*, 6(1):3–14, 1999.

[2] D. F. Andrews. Plots of high dimensional data. *Biometrics*, 28:125–136, 1972.

[3] B. Baldi and K. Hornik. Neural networks and principal component analysis: Learning from examples without local minima. *Neural Networks*, 2:53–58, 1989.

[4] R. E. Barlow, D. J. Bartholomew, J. M. Bremmer, and H. D. Brunk. *Statistical Inference under Order Restrictions*. Wiley, New York, 1972.

[5] Anna Bartkowiak and Adam Szustalewicz. Some modern techniques for viewing multivariate data - a comparative look. In M. A. Kłopotek and M. Michalewicz, editors, *Workshop on Intelligent Information Systems VIII, Ustroń, Poland*, pages 7–11, June 14-18 1999.

[6] Wojciech Basalaj. Incremental multidimensional scaling method for database visualization. In *Proceedings of Visual data Exploration and Analysis VI, SPIE*, volume 3643, pages 149–158, San Jose, California, USA, January 1999.

[7] Hans-Ulrich Bauer, R. Der, and M. Herrman. Controlling the magnification factor of self-organizing feature maps. *Neural Computation*, 8:757–771, 1996.

[8] Hans-Ulrich Bauer and Klaus K. Pawelzik. Quantifying the neighborhood preservation of self-organizing feature maps. *IEEE Transaction on Neural Networks*, 3(4):570–579, 1992.

[9] Hans-Ulrich Bauer and Thomas Willmann. Growing a hypercubical output space in a self-organizing feature map. *IEEE Transaction on Neural Networks*, 8(2):218–226, 1997.

[10] Jean-Pierre Benzécri. *L'analyse des données Vol. 2: L'analyse des correspondances*. Dunod, Paris, 1973.

[11] James C. Bezdek and Nikhil R. Pal. An index of topological preservation for feature extraction. *Pattern Recognition*, 28(3):381–391, 1995.

[12] Adam Biela. *Skalowanie wielowymiarowe w analizach ekonomicznych i behawioralnych*. Norbertinum, Lublin, 1995.

[13] Bruno Bienfait and Johann Gasteiger. Spinne. *Journal of molecular graphics and modeling*, 1997.

[14] Christopher M. Bishop, Markus Svensén, and Christopher K. I. Williams. GTM: A principled alternative to the self-organizing map. Technical report, Aston University, Birmingham, Neural Computing Research Group, April 1996. Available as NCRG/96/015 from `http://www.ncrg.aston.ac.uk/`.

[15] Gautam Biswas, Anil K. Jain, and Richar C. Dubes. Evaluation of projection algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-3(6):701–708, November 1982.

[16] C.L. Blake and C.J. Merz. UCI repository of machine learning databases. University of California, Irvine, Dept. of Information and Computer Sciences, 1998. `http://www.ics.uci.edu/∼mlearn/MLRepository.html`.

[17] Leon Bobrowski and Magdalena Topczewska. Designing of linear visualising transformations for exploration of databases. In Leon Bobrowski, Jan Doroszewski, Etore Marubini, and Norbert Victor, editors, *Statistics and clinical practice*, volume 50 of *Lecture notes of the ICB seminars*, pages 15–19. Polska Akademia Nauk, Międzynarodowy Centrum Biocebernetyki, Warsaw, June 2000.

[18] S. A. Boorman and H. C. White. Social structure from multiple networks, II. Role structures. *American Journal of Sociology*, 81:1384–1446, 1976.

[19] Andreas Buja and Daniel Asimov. Grand tour methods: An outline. In *Computer Science and Statistics*, Proceedings of the 17-th Symposium on the Interface, pages 63–67, 1986.

[20] J.N. Butcher, J.R. Graham, C.L. Williams, and Y.Ben Pooth. *Development and use for the MMPI-2. Content Scales.* University of Minnesota Press, Minneapolis.

[21] J.N. Butcher and C.L. Williams. *Essentials of MMPI-2 and MMPI-A interpretation*. University of Minnesota Press, Minneapolis, 1992.

[22] J. D. Carroll and P. Arabie. Multidimensional scaling. *Ann. Rev. Psychol.*, 31:607–49, 1980.

[23] C. L. Chang and R. C. T. Lee. A heuristic relaxation method for nonlinear mapping in cluster analysis. *IEEE Transactions on Computers*, pages 197–200, March 1973.

[24] H. Chernoff. Using faces to represent points in *k*-dimensional space graphically. *Journal of the American Statistical Association*, 68(342):361–368, 1973.

[25] C. H. Coombs. *A theory of data*. John Wiley & Sons, New York, 1964. Chapters 5, 6 and 7, pp. 80-180.

[26] Marie Cottrell and Eric de Bodt. A Kohonen map representation to avoid misleading interpretations. In *Proceedings of the European Symposium on Artificial Neural Networks*, pages 103–110, Bruges, 1996.

[27] Trevor F. Cox and Michael A. A. Cox. *Multidimensional Scaling*, volume 59 of *Monographs on Statitics and Applied Probability*. Chapman & Hall, London, 1994.

[28] A. P. M. Coxon and C. L. Jones. *Multidimensional scaling*, volume 1: Exploring Data Structures, pages 296–339. C. A. O'Muircheartaigh, C. Payne. Wiley, New York, 1977.

[29] A.P.M. Coxon. *The user's guide to MDS*. Heinemann Educational Books Ltd, 1982.

[30] S. De Backer, A. Naud, and P. Scheunders. Non-linear dimensionality reduction techniques for unsupervised feature extraction. *Pattern Recognition Letters*, 19:711–720, 1998.

[31] J. A. de Leeuw. *Recent developments in statistics*, chapter Applications of convex analysis to multidimensional scaling, pages 133–145. Amsterdam: North Holland, 1977.

[32] Jan de Leeuw and Willem Heiser. *Handbook of statistics*, volume 2, chapter Theory of multidimensional scaling, pages 285–316. North-Holland, krishnaiah P.R. and Kanal L.N. edition, 1982.

[33] Guido Deboeck and Teuvo Kohonen. *Visual exploration in finance with self-organising maps*. Springer Finance, 1998.

[34] P. Demartines. *Analyse de données par réseaux de neurones auto-organisés*. PhD thesis, Institut National Polytechnique de Grenoble, November 1994. 202 pages, 84 figures, 6 pages of index, 123 references.

[35] P. Demartines and J. Hérault. Curvilinear component analysis: A self-organizing neural network for nonlinear mapping of data sets. *IEEE Transaction on Neural Networks*, 8(1):148–154, January 1997.

[36] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, B*, 39(1):1–38, 1977.

[37] T. Deutscher. *Issues in data collection and reliability in marketing MDS studies - Implications for large stimulus sets*. Colledge, R.G., Rayner, J.N., eds., 1980.

[38] J. K. Dixon. Pattern recognition with partly missing data. *IEEE Transaction on Systems, Man and Cybernetics*, 9:617–621, 1979.

[39] Włodzisław Duch. Quantitative measures for the self-organizing topographic maps. *Open Systems & Information Dynamics*, 1995.

[40] Włodzisław Duch and Antoine Naud. Multidimensional scaling and Kohonen's self-organizing maps. In *Proceedings of the Second Conference "Neural Networks and their Applications"*, volume I, pages 138–143, Szczyrk, 1996.

[41] Włodzisław Duch and Antoine Naud. On global self-organizing maps. In *Proceedings of the "4th European Symposium on Artificial Neural Networks"*, pages 91–96, Bruges, 1996.

[42] Włodzisław Duch and Antoine Naud. Simplexes, multi-dimensional scaling and self-organized mapping. In *Proceedings of the "8th joint EPS-APS International Conference on Physics Computing '96"*, pages 367–370, Kraków, 17-21.9 1996.

[43] O. R. Duda and E. P. Hart. *Pattern classification and scene analysis*. John Wiley, New York, 1973.

[44] Witold Dziwnel. In search for the global minimum in problems of features extraction and selection. In *Proceedings of the EUFIT'95*, Aachen, Germany, August 1995.

[45] E. Erwin, K. Obermayer, and K. Schulten. Self-organizing maps: ordering, convergence properties and energy functions. *Biological Cybernetics*, 67:47–55, 1992.

[46] B. Everitt. *Graphical techniques for multivariate data*. New York, North-Holland, 1978.

[47] R.A. Fisher. The use of multiple measurements on taxonomics problems. *Annals of Eugenics*, pages 179-188, 1936.

[48] John A. Flanagan. Self-organisation in SOM. *Neural networks*, 9(7):1185–1197, 1996.

[49] George E. Forsythe, Michael A. Malcolm, and Cleve B. Moler. *Least squares and the singular value decomposition*. Prentice Hall, 1977.

[50] Jerome H. Friedman and John W. Tukey. A projection pursuit algorithm for exploratory data analysis. *IEEE Transactions on Computers*, C-23(9):881–890, September 1974.

[51] Bernd Fritzke. Growing cell structure – A self-organizing network for unsupervised and supervised learning. Technical report, International Computer Science Institute, May 1993.

[52] Keinosuke Fukunaga. *Introduction to statistical pattern recognition*. Computer Science and Scientific Computing. Academic Press, second edition, 1990.

[53] Colin Fyfe. *Artificial Neural Networks and Information Theory*. Department of Computing and Information Systems, University of Paisley, first edition, 1996.

[54] K. R. Gabriel. The biplot graphic display of matrices with application to principal components analysis. *Biometrika*, 58(3):453–467, 1971.

[55] R. T. G. Golledge and A. N. Spector. Comprehending the urban environment: Theory and practice. *Geogr. Anal.*, 10:403–26, 1978.

[56] G. H. Golub and C. Reinsch. *Handbook for automatic computation*, volume II - Linear algebra, chapter Singular Value Decomposition and least squares solutions, pages 134–151. Springer Verlag, 1971.

[57] Gene H. Golub and Charles F. Van Loan. *Matrix computations*. John Hopkins series in the mathematical sciences. The Johns Hopkins University Press, second edition, 1989.

[58] G. J. Goodhill, S. Finch, and T. J. Sejnowski. Quantifying neighbourhood preservation in topographic mappings. Technical Report INC-9505, Institute for Neural Computation, November 1995.

[59] Josef Göppert and Wolfgang Rosenstiel. Interpolation in SOM: Improved generalization by iterative methods. In EC2 & Cie, editor, *Proceedings of International Conference on Artificial Neural Networks ICANN 95*, Paris, France, October 1995.

[60] J. C. Gower. Some distance properties of latent root and vector methods used in multivariate analysis. *Biometrika*, 53:325–38, 1966.

[61] J. C. Gower and D. J. Hand. *Biplots*. Chapman & Hall, 1996.

[62] J. C. Gower and Ross G. J. S. Minimum spanning trees and single linkage cluster analysis. *Applied Statistics*, 18:54–64, 1969.

[63] Michael J. Greenacre. *Theory and applications of correspondence analysis*. Academic Press, London, 1984.

[64] Patrick Groenen and Willem J. Heiser. The tunneling method for global optimization in multidimensional scaling. *Psychometrika*, 61(3):529–550, September 1996.

[65] S. Haykin. *Neural networks: A comprehensive foundation*. MacMillan College Publishing, New York, 1994.

[66] D. O. Hebb. *The organisation of behavior*. New York, Wiley, 1949.

[67] W. J. Heiser. A generalized majorization method for least square multidimensional scaling of pseudodistances that may be negative. *Psychometrika*, 56(1):7–27, 1991.

[68] Hoffman and Joachim M. Buhmann. Multidimensional scaling and data clustering. In G. Tesauro, D. S. Touretzky, and T. K. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7, pages 459–460. Cambridge Mass. MIT Press, 1995.

[69] Timo Honkela. *Self-organizing maps in natural language processing*. PhD thesis, Helsinki University of Technology, 1997.

[70] Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24(6):417–441, 1933.

[71] J. Iivarinen, T. Kohonen, J. Kangas, and S. Kaski. Visualizing the clusters on the self-organizing map. In Tapio Reponen Christer Carlsson, Timo Järvi, editor, *Proceedings of the Conference on Artificial Intelligence Res. in Finland*, volume 12, pages 122–126, Helsinki, Finland, 1994. Finnish Artificial Intelligence Society. Multiple Paradigms for Artificial Intelligence (SteP94).

[72] N. Jankowski. *Ontogenic neural networks and their applications to classification of medical data*. PhD thesis, Department of Computer Methods, Nicholas Copernicus University, Toruń, Poland, 1999.

[73] Samuel Kaski. *Data exploration using self-organising maps*. PhD thesis, Helsinki University of Technology, 1997.

[74] Samuel Kaski, Jaari Kangas, and Teuvo Kohonen. Bibliography of self-organising maps (SOM) papers: 1981-1998. Technical report, Neural Computing Surveys, 1998.

[75] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[76] R. W. Klein and R. C. Dubes. Experiments in projection and clustering by simulated annealing. *Pattern Recognition*, 22(2):213–220, 1989.

[77] Hansjörg Klock and Joachim M. Buhmann. Multidimensional scaling by deterministic annealing. In Springer Lecture Notes in Computer Science Venice, editor, *Proceedings of the International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition, EMMCVPR'97*, volume 1223, pages 246–260, May 1997.

[78] Teuvo Kohonen. Automatic formation of topological maps of patterns in a self-organizing system. In Erkki Oja and Olli Simula, editors, *Proceedings of Second Scandinavian Conference on Image Analysis, Espoo, Finland*, pages 214–220, 1981.

[79] Teuvo Kohonen. *Statistical pattern recognition revisited*. Elsevier Science Publishers, North Holland, 1990.

[80] Teuvo Kohonen. *Self-Organizing Maps*. Springer-Verlag, Heidelberg Berlin, 1995.

[81] Teuvo Kohonen, Jussi Hynninen, Jari Kangas, and Jorma Laaksonen. *SOM_PAK The self-organizing map program package Version 3.1*. Helsinki University of Technology, Laboratory of Computer and Information Science Rakentajanaukio 2 C, SF-02150 Espoo, April 7 1995. Available at `ftp://cochlea.hut.fi/pub/som_pak`.

[82] Teuvo Kohonen and Helge Ritter. Self-organizing semantic maps. *Biological cybernetics*, 61:241–254, 1989.

[83] M. A. Kraaijveld, J. Mao, and A. K. Jain. A nonlinear projection method based on Kohonen's topology preserving maps. *IEEE Transactions on neural networks*, 6(3):548–559, 1995.

[84] M. A. Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal*, 37(2):233–243, 1991.

[85] Joseph B. Kruskal. Non metric multidimensional scaling : a numerical method. *Psychometrika*, 29:115–129, 1964.

[86] Joseph B. Kruskal. Comments on ”A nonlinear mapping for data structure analysis”. *IEEE Trans. on computers*, page 1614, December 1971.

[87] Joseph B. Kruskal. *Multidimensional scaling and other methods for discovery structure*, volume III, pages 296–339. John Wiley, New York, 1977.

[88] Joseph B. Kruskal and J. Douglas Carroll. Geometrical models and badness-of-fit functions. In Academic Press Pachuri, R. Krishnaiah, editor, *Multivariate Analysis II*, pages 103–110, 1969.

[89] Joseph B. Kruskal and Myron Wish. *Multidimensional Scaling*. Sage Publications, Beverly Hills, CA, 1978.

[90] Joseph B. Kruskal, Forrest W. Young, and Judith B. Seery. *How to use KYST A very flexible program to do multidimensional scaling and unfolding*. ATT Bell Labs., 1977. Fortran software.

[91] D. N. Lawley and A. E. Maxwell. *Factor Analysis as a Statistical Method*. Butterworth & Co., London, second edition, 1971.

[92] R. C. T. Lee, J. R. Slagle, and H. Blum. A triangulation method for the sequential mapping of points from n-space to two-space. *IEEE Transactions on Computers*, pages 288–292, March 1977.

[93] S. Maital. Multidimensional scaling: Some econometric applications. *Journal of Econometrics*, 8:33–46, 1978.

[94] O. L. Mangasarian and W. H. Wolberg. Cancer diagnosis via linear programming. *SIAM News*, 23(5):1 & 18, September 1990.

[95] J. Mao and A. K. Jain. Artificial neural networks for feature extraction and multivariate data projection. *IEEE Transaction on Neural Networks*, 6(2):296–317, March 1995.

[96] T. Martinetz and K. Schulten. A neural gas network learns topologies. In T. Kohonen *et al.*, editor, *IEEE International Conference on Artificial Neural Networks, Espoo, Finland*, volume 1, pages 397–407. Elsevier, 1991.

[97] W. S. McCulloch and W. Pitts. A logical calculus of ideas immanent in neural activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.

[98] D. Michie, D. J. Spiegelhalter, and C. C. Taylor, editors. *Machine Learning, Neural and Statistical Classification*. Ellis Horwood, New York, 1994.

[99] Filip Mulier and Vladimir Cherkassky. Learning rate schedules for self-organising maps. In *Proceedings of the 12th International Conference on Pattern recognition IAPR, Jerusalem, Israel*, volume II, pages 224–228, 1994.

[100] Antoine Naud. Application of Kohonen's Self-Organizing Maps to textured image segmentation. In *Proceedings of the Symposium "System Modeling Control 8"*, volume 3: Artificial Neural Networks and their applications, pages 91–95, Zakopane, 1995.

[101] Antoine Naud and Włodzisław Duch. Interactive data exploration using MDS mapping. In *Proceedings of the Fifth Conference "Neural Networks and Soft Computing"*, pages 255–260, Zakopane, Poland, 2000.

[102] Heinrich Niemann and Jürgen Weiss. A fast-converging algorithm for nonlinear mapping of high-dimensional data to a plane. *IEEE Transactions on Computers*, C-28(2):142–147, February 1979.

[103] E. Oja. A simplified neuron model as a principal component analyzer. *Journal of Mathematical Biology*, 16:267–273, 1982.

[104] E. Oja. Neural networks, principal components and subspaces. *International Journal of Neural Systems*, 1:61–68, 1989.

[105] E. Oja. Principal components, minor components and linear neural networks. *Neural Networks*, 5:927–935, 1992.

[106] R. W. Olshavsky, D. B. MacKay, and G. Sentell. Perceptual maps of supermarket location. *J. Appl. Psychol.*, 60:80–86, 1975.

[107] Karl Pearson. On lines and planes of closest fit to system of points in space. *Phil. Mag.*, 2(11):559–572, 1901.

[108] Elżbieta Pekalska, Dick de Ridder, Robert P.W. Duin, and Martin A. Kraaijveld. A new method of generalizing sammon mapping with application to algorithm speed-up. In M. Boasson, J.A. Karndorp, JFM Torino, and MG. Vosselman, editors, *ASCI'99 Proc. 5th Annual Conference of the Advanced School for Computing and Image*, pages 221–228, Heijen, NL, June 1999.

[109] Vadim Pliner. Metric unidimensional saling and global optimization. *Journal of classification*, 13:3–18, 1996.

[110] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C*. Cambridge University Press, Cambridge, second edition, 1992.

[111] J. O. Ramsay. Some statistical approaches to multidimensional scaling data. *J. R. Statist. Soc. A*, 145(Part. 3):285–312, 1982.

[112] B. D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, Cambridge. England, 1996.

[113] H. Ritter and K. Schulten. Convergence properties of Kohonen's topology conserving mappings: Fluctuations, stability and dimension selection. *Biological Cybernetics*, 60:59–71, 1988.

[114] F. Rosenblatt. The perceptron - a perceiving and recognizing automaton. Technical Report 85-460-1, Cornell Aeronautical Laboratory, 1957.

[115] J. W. Sammon. A nonlinear mapping for data analysis. *IEEE Transactions on Computers*, C-18(5):401–409, 1969.

[116] T. D. Sanger. Optimal unsupervised learning in a single-layer linear feedforward neural network. *Neural Networks*, 2(6):459–473, 1989.

[117] P. Scheunders, S. De Backer, and A. Naud. Non-linear mapping for feature extraction. In *Proceedings of the Joint IAPR International Workshops SSPR'98 and SPR'98*, volume 1451 of *Lecture Notes in Computer Science*, pages 823–830, Sydney, Australia, August 1998. Springer.

[118] Roger N. Shepard. The analysis of proximities: Multidimensional scaling with an unknown distance function. I. *Psychometrika*, 27(2):125–140, 1962.

[119] Roger N. Shepard. Metric structures in ordinal data. *Journal of Mathematical Psychology*, 3:287–315, 1966.

[120] Wojciech Siedlecki, Kinga Siedlecka, and Jack Sklansky. Experiments on mapping techniques for exploratory pattern analysis. *Pattern Recognition*, 21(5):431–438, 1988.

[121] Wojciech Siedlecki, Kinga Siedlecka, and Jack Sklansky. An overview of mapping techniques for exploratory pattern analysis. *Pattern Recognition*, 21(5):411–429, 1988.

[122] H. Speckmann, G. Raddatz, and W. Rosenstiel. Relations between generalized fractal dimensions and Kohonen's self-organizing maps. In *Journées Neurosciences et Sciences de l'Ingénieur*, Paris, May 1994.

[123] Johan Fredrik Markus Svensén. *GTM: The generative topographic mapping*. PhD thesis, Aston University, April 1998.

[124] Deborah F. Swayne, Dianne Cook, and Andreas Buja. XGobi: Interactive dynamic data visualization in the X Window system. *AT&T Labs - Research*, 1998. Available at `http://www.research.att.com/∼andreas/xgobi/`.

[125] Ryszard Tadeusiewicz. *Sieci neuronowe*. Problemy Współczesnej Nauki i Techniki: Informatyka. Akademicka Oficyna Wydawnicza RM, Warszawa, 1993.

[126] Y. Takane, F. W. Young, and J. de Leeuw. Nonmetric individual differences multidimensional scaling: an alternating least square method with optimal scaling features. *Psychometrika*, 42:7–67, 1977.

[127] Yoshio Takane. Multidimensional successive categories scaling: A maximum likelihood method. *Psychometrika*, 46(1):9–28, March 1981.

[128] Patrick Thiran and Martin Hasler. Self-organisation of a one-dimensional Kohonen network with quantized weights and inputs. *Neural networks*, 9(7):1185–1197, 1996.

[129] Michael E. Tipping. *Topographic mappings and feed-forward neural networks*. PhD thesis, The University of Aston in Birmingham, February 1996.

[130] W. Tobler and S. Wineburg. A cappadocian speculation. *Nature*, 231:39–41, 1971.

[131] W. S. Torgerson. Multidimensional scaling. 1. Theory and method. *Psychometrika*, 17:401–419, 1952.

[132] W. S. Torgerson. *Theory and methods of scaling*. John Wiley & Sons, New York, 1958.

[133] Takahiro Tsuchiya. A probabilistic multidimensional scaling with unique axes. *Japanese Psychological Research*, 38(4):204–211, 1996.

[134] J. W. Tukey. *Exploratory data analysis*. Addison-Wesley, Reading MA, 1977.

[135] A. Ultsch. Self-organized feature maps for monitoring and knowledge aquisition of a chemical process. In S. Gielen and B. Kappen, editors, *Proceedings of the International Conference on Artificial Neural Networks (ICANN93)*, pages 864–867. Springer-Verlag London, 1993.

[136] M. C. van Wezel, J. N. Kok, and W. A. Kosters. Two neural network methods for multidimensional scaling. In *Proceedings of the European Symposium on Artificial Neural Networks (ESANN'97)*, pages 97–102, Bruges, Belgium, 1997.

[137] Juha Vesanto. SOM-based data visualization methods. *Intelligent Data Analysis*, 3(2):111–126, 1999.

[138] Da-Kai Wang, Roderick B. Urquhart, and James E. S. MacLeod. The equal angle spanning tree mapping: A sequential method for projecting from h-space to 2-space. *Pattern Recognition Letters*, 2:69–73, 1983.

[139] A. R. Webb. Multidimensional scaling by iterative majorization using radial basis functions. *Pattern Recognition*, 28(5):753–759, 1995.

[140] H. F. Weisberg. Scaling models for legislative roll-call analysis. *Am. Polit. Sci. Rev.*, 66:1306–15, 1972.

[141] B. Widrow and M. E. Hoff. Adaptive switching circuits. In *IRE WESCON Convention Record*, volume 4, pages 96–104. Reprinted in Anderson and Rosenfeld (1988), 1960.

[142] G. Young and A. S. Householder. Discussion of a set of points in terms of their mutual distances. *Psychometrika*, 3(1):19–22, March 1938.