3

MONOGRAPH SERIES: S.T. WIERZCHOŃ, M.A. KŁOPOTEK ALGORITHMS OF CLUSTER ANALYSIS

3

MONOGRAPH SERIES

SŁAWOMIR T. WIERZCHOŃ
MIECZYSŁAW A. KŁOPOTEK

ALGORITHMS OF CLUSTER ANALYSIS

INSTITUTE OF COMPUTER SCIENCE
POLISH ACADEMY OF SCIENCES

MONOGRAPH SERIES
INFORMATION TECHNOLOGIES: RESEARCH
AND THEIR INTERDISCIPLINARY APPLICATIONS

3

SŁAWOMIR T. WIERZCHOŃ
MIECZYSŁAW A. KŁOPOTEK

ALGORITHMS OF CLUSTER ANALYSIS

INSTITUTE OF COMPUTER SCIENCE
POLISH ACADEMY OF SCIENCES

Warsaw, 2015

Publication issued as a part of the project:
'Information technologies: research and their interdisciplinary applications',
Objective 4.1 of Human Capital Operational Programme.
Agreement number UDA-POKL.04.01.01-00-051/10-00.

Publication is co-financed by European Union from resources of European Social Fund.

Project leader: Institute of Computer Science, Polish Academy of Sciences

Project partners: System Research Institute, Polish Academy of Sciences, Nałęcz Institute of Biocybernetics and Biomedical Engineering, Polish Academy of Sciences

Editors-in-chief: Olgierd Hryniewicz
Jan Mielniczuk
Wojciech Penczek
Jacek Waniewski

Reviewer: dr hab. inż. Jerzy Rudolf Tchórzewski, prof. UPH

Sławomir T. Wierzchoń
Institute of Computer Science, Polish Academy of Sciences
Slawomir.Wierzchon@ipipan.waw.pl
http://www.ipipan.waw.pl/staff/s.wierzchon

Mieczysław A. Kłopotek
Institute of Computer Science, Polish Academy of Sciences
Mieczyslaw.Klopotek@ipipan.waw.pl
http://www.ipipan.waw.pl/staff/m.klopotek/

Publication is distributed free of charge

ISBN 978-83-63159-10-8
e-ISBN 978-83-63159-11-5

Layout: Piotr Borkowski
Cover design: Waldemar Słonina

Contents

List of Figures

List of Tables

List of Algorithms

List of most important symbols

$\mathfrak{X}$	set of objects (observations)
$\mathfrak{x}$	an element of the set $\mathfrak{X}$
$X \in \mathbb{R}^{m\times n}$	matrix, representing the set of objects, $X = (\mathbf{x}_1 \ \cdots \ \mathbf{x}_m)^{\mathsf{T}}$
m	cardinality of the set X
n	dimensionality (number of coordinates) of the vector of features of $\mathbf{x} \in X$
C	set of clusters
k	number of clusters

$\mathrm{diag}(\mathbf{v})$	diagonal matrix having the main diagonal equal to the vector $\mathbf{v}$

$G = (V, E)$	graph spanned over the set of vertices V linked with the edges indicated in the set E
$N(v)$	set of neighbours of the vertex v in graph G
$N_k(v)$	set of k nearest neighbours of the vertex v in graph G
$\partial(C)$	edge separator in graph G: $\partial(C) = \{\{u, v\} \in E : u \in C, v \notin C\}$
A	neighbourhood matrix having elements $a_{ij} \in \{0, 1\}$, $A = [a_{ij}]_{m\times m}$
S	similarity matrix having elements $s_{ij} \in [0, 1]$, $S = [s_{ij}]_{m\times m}$
$\mathfrak{m}$	number of edges in graph G
d_i	degree (weight) of the i-th object, $\mathsf{d}_i = \sum_{j=1}^{m} s_{ij}$
D	matrix of degrees, $D = diag([\mathsf{d}_1, \dots, \mathsf{d}_m])$
L	combinatorial Laplacian, $L = D - S$
$\mathfrak{L}$	normalised Laplacian, $\mathfrak{L} = D^{-1/2} L D^{-1/2}$
$\mathbb{L}$	a variant of the Laplacian, $\mathbb{L} = \mathbb{I} - P$
P	column-stochastic transition matrix, describing walk on graph G, $P = SD^{-1}$
$\widetilde{P}$	row-stochastic transition matrix, describing walk on graph G, $\widetilde{P} = D^{-1}S$
$\widehat{P}$	column-stochastic transition matrix in a lazy walk on graph G, $\widehat{P} = \frac{1}{2}(\mathbb{I} + SD^{-1})$
$\boldsymbol{\pi}$	stationary distribution of the Markov chain having transition matrix P, that is, a vector such that $\boldsymbol{\pi} = P\boldsymbol{\pi}$
$\rho(s, \alpha)$	global PageRank with positive preference vector s and the coefficient of teleportation α: $\rho(s, \alpha) = \alpha s + (1 - \alpha)P\rho(s, \alpha)$
$\mathfrak{p}(s, \alpha)$	personalised PageRank vector with non-negative preference vector s and the coefficient of teleportation α

$\mathbf{e}$	column vector having elements equal 1
$\mathbb{I}$	unit matrix

1 Introduction

The role of grouping is to divide the set of objects into homogeneous groups: two arbitrary objects belonging to the same group are more similar to each other than two arbitrary objects belonging to different groups. If we wish to apply this recipe in practice, we must find the answers to two basic questions: (a) how to define the similarity between the objects, and (b) in what manner should one make use of the thus defined similarity in the process of grouping. The fact that the answers to these questions are provided independently one from another results in the multiplicity of algorithms

The methods of data analysis can be roughly classified into two categories[1]: (a) the descriptive (exploratory) ones, which are recommended when, having no initial models or hypotheses, we try to understand the general nature and structure of the high-dimensional data, and (b) the confirmatory (inferential) methods, applied in order to confirm the correctness of the model or of the working hypotheses, concerning the data collected. In this context, a particular role is played by various statistical methods, such as, for instance, analysis of variance, linear regression, discriminant analysis, multidimensional scaling, factor analysis, or, finally, the subject of our considerations here - cluster analysis[2] [213], [164].

It is really difficult to list all the domains of theoretical and practical applications of cluster analysis. Jain [190] mentions three basic domains: image segmentation [191], information retrieval [246], and bio-informatics [34].

An interesting example of application of cluster analysis in modern information retrieval is constituted by the `CLUSTY` search engine (`clusty.com`). The methods of cluster analysis play an important role in the development of the recommender systems [180], [300], in various kinds of economic, medical etc. analyses. Application of these methods results from at least three reasons:

(a) To gain insight into the nature of the data, and, first of all, to indicate the typical and the atypical (*outlying*) data items, to uncover the potential anomalies, to find hidden features, or, finally - to be able to formulate and verify the hypotheses concerning the relations between the observations;
(b) To obtain a compact description of data, to select the most representative objects; here, the classical application is image compression;

[1] See, e.g., J. W. Tukey, *Exploratory Data Analysis.* Addison-Wesley, 1977.
[2] This area is also referred to as Q-analysis, typology, clumping, and taxonomy, [191], depending upon the domain of application.

(c) To obtain a natural classification of data, for instance - by determining the similarities between pairs of objects to establish the respective hierarchical structures.

Methods of cluster analysis are being applied there, where we wish to understand the nature of the phenomenon, represented by the set of observations, to get a sort of summary of the content of large data sets, and to process such sets effectively.

Jain and Dubes [191] list the following challenges, linked with the practical use of clustering:

(a) What is a cluster (group, module)?
(b) What features ought to be used to analyse the data collected?
(c) Should the data be normalised?
(d) Are there outliers in the analysed dataset, and if so - how should they be treated?
(e) How to define the similarity between the pairs of objects?
(f) How many clusters are really there in the data set? Do we deal, at all, with clusters?
(g) What method should be used in a given situation?
(h) Is the obtained partition of data justified?

Some of these questions find at least partial answers in a multiplicity of books, devoted to various aspects of cluster analysis, like [10], [51], [112], [191], [6], or [121].

The recent years, though, brought a number of new challenges. First, cluster analysis is being applied nowadays to processing of huge sets of data, which makes it necessary to develop specialised algorithms. Second, the necessity of analysing the data sets having complex topologies (data being situated in certain submanifolds) leads to the requirement of applying more refined tools. We mean here the spectral methods, the methods making use of kernel functions, and the relations between these two groups of methods.

The starting point for these methods is constituted by the matrix, describing the strength of interrelations between the objects making up the data set. In the case of the kernel-based methods, considerations are being transferred to the highly dimensional and nonlinear space of features for the purpose of strengthening of the separability of data. Due to application of the so-called *kernel trick* to determine the distance between objects in this new space, the knowledge of a certain kernel function suffices. The elements of the matrix, mentioned before, are the values of the kernel function, calculated for the respective pairs of objects. On the other hand, in the case of spectral methods, the elements of the matrix correspond to the values of similarity of the pairs of objects. Eigenvalues and eigenvectors of this matrix, or a matrix, being its certain transformation (usually a form of the Laplacian of the graph, corresponding to the similarity matrix), provide important information on relations between objects.

An interesting example is constituted by the problem of ordering of documents, collected by the crawlers, cooperating with a search engine. We deal in such cases with enormous collections of objects (of the order of 10^9), with the issue of their effective representation, and, finally, with the task of indicating the important documents. The common sense postulate that the important websites (documents) are the ones that are referred to (linked to) by other important websites allows for the reformulating of the problem of assigning ranks in the ordering into the problem of determining the dominating eigenvector of the stochastic matrix P, being a slight modification of the original matrix A, representing connections between the websites, that is $a_{ij} = 1$ when on the i^{th} website there is a link to the j^{th} one. Vector $\mathbf{r}$, which satisfies the matrix equation $\mathbf{r} = P^{\mathrm{T}}\mathbf{r}$, is called the PageRank vector [274]. Both this vector and its diverse variants (TotalRank, HillTop, TrustRank, GeneRank, IsoRank, etc.) are the examples of the so-called spectral ranking, which find application not only in the ordering of documents (or, more generally, in various aspects of bibliometrics), but also in graph theory (and, consequently, also in the analysis of social networks), in bio-informatics, and so on.

A development over the latter idea is constituted by the methods based on random walk on graphs. Namely, the task of grouping is here considered as an attempt of finding such a division of the graph that a randomly moving wanderer shall remain for a long time in a single cluster (subgraph), only rarely jumping between the clusters. An interesting work on this subject is presented in, e.g., [146], while a number of theoretical results have been shown by Chung [75], [77], [14], and by Meila [252].

The transformation of the matrices, representing relations (e.g. similarity) between the elements of the set of objects, into the stochastic Markovian matrices leads to yet another interesting formalism. As we treat the eigenvectors of these matrices as forming a new system of coordinates, we transform the multidimensional data into a cloud of points in a space with a much lower number of dimensions. This new representation reflects the essential properties of the original multidimensional structure. In this manner we enter the domain of the so-called diffusion maps [83], which have two significant features, distinguishing them from the classical methods: they are nonlinear, and, besides this, they do faithfully map the local structure of data. From the point of view of, for instance, processing of text documents, an essential advantage of such an approach is a relatively straightforward adaptation to the semi-supervised clustering problems and the *coclustering* problems - the important, quite recently developing, directions of machine learning.

These most up-to-date trends in grouping have not been, until now, systematically presented nor considered in an integrated manner against the background of the classical methods of cluster analysis. Such a survey would be very welcome indeed from the point of view of students with an interest in the techniques of knowledge extraction from data and computer-based modelling, in the context of their application in pattern recognition, signal analysis, pattern and interdependence identification, and establishment of other interesting characteristics in the large and very large collections of data, especially those put together automatically in real time.

That is why we prepared this monograph, which:

(a) Contains the survey of the basic algorithms of cluster analysis, along with presentation of their diverse modifications;
(b) Makes apparent the issue of evaluation and proper selection of the number of clusters;
(c) Considers the specialisation of selected algorithms regarding the processing of enormous data sets;
(d) Presents in a unified form a homogeneous material, constituting the basis for developing new algorithms of spectral and diffusion data analysis;
(e) Comments upon the selected solutions for the semi-supervised learning;
(f) Provides a rich bibliography, which might also be a starting point to various individual research undertakings.

The limited volume of this book did not allow to cover all the developments in the field. On the one hand, we do not discuss algorithms elaborated for theoretical purposes of clusterability theory. On the other hand, we do not consider algorithms developed for particular applications only, like medical imaging. We concentrated on the clustering paradigm of a cluster as group of objects that are more similar to one another than to objects from other clusters. But other concepts are possible, based e.g. on classification/regression capabilities. Some researchers consider clusters as sets of objects where the attributes are (more) easily predictable from one another than in general population (mutual prediction). We mention only in passing that one may seek clusters either in a given space or in its subspaces (projections of the original space), but this is by itself a separate research area, answering questions what is the optimal number of features and which subspace (possibly linearly transformed) should be used. Still another research area concentrates around the so-called optimal modularity, where we seek clusters in which similarity pattern does not occur to be random. Still another area of investigation are special forms of optimised cluster quality function which ensure better convergence, like sub-modular functions.

2

Cluster Analysis

Cluster analysis consists in distinguishing, in the set of analysed data, the groups, called clusters. These groups are disjoint[1] subsets of the data set, having such a property that data belonging to different clusters differ among themselves much more than the data, belonging to the same cluster. The role of cluster analysis is, therefore, to uncover a certain kind of natural structure in the data set. The means enabling performing that task is constituted usually by a certain measure of similarity or dissimilarity – the issue is discussed further on in Section 2.2. Cluster analysis is not only an important cognitive tool, but, as well, a method for reducing large sets of data, since it allows for the replacement of a group of data by its compact characterisation, like, e.g. the centre of gravity of the given group.

The task of cluster analysis can be perceived as a problem of grouping of objects according to their mutual similarity. Objects, which are mutually similar in a sufficiently high degree, form a homogeneous group (a cluster). It is also possible to consider the similarity of objects to certain characteristic entities (called prototypes) of the classes. In this case we deal more with the problem of classification, that is - of finding the model patterns - see [112]. Yet, if the characteristics corresponding to classes are not given a priori, they should be established. And this is exactly what cluster analysis is about.

The term *data clustering* (meaning grouping of data) appeared for the first time in 1954 in the title of a paper concerning the analysis of anthropological data, [190, p.653]. Other equivalent names, given cluster analysis, are *Q-analysis*, *typology*, *clumping*, and *taxonomy* [191], depending on the domain, in which clustering is applied. There is a number of very good books, devoted to cluster analysis. The classical ones include[2]: [10], [112], [121], [191], [201], [319], [337]. Among the more recent and specialised monographs we can mention:[5], [55], [51], [54], [93], [209], [280], [289], [332]. A reader, interested in survey works may wish to have a look at, e.g., [47], [192], [190], [373]. More advanced techniques of cluster analysis are considered, in particular, in [124].

[1] The requirement of disjoint subsets is used in the classical data analysis. In the general case, the groups distinguished might constitute the *coverage* of the data set. This is the case, for instance, in the fuzzy data analysis.

[2] Many of those listed have been modified several times over and successive editions have been published.

2.1 Formalising the problem

It is usually assumed in cluster analysis, that a set of m objects $\mathfrak{X} = \{\mathfrak{x}_1, \dots, \mathfrak{x}_m\}$ is given, with every object being described by an n-dimensional vector $\mathbf{x}_i = (x_{i1} \dots x_{in})^{\mathsf{T}}$, where x_{ij} denotes the value of the j-th feature of the object $\mathfrak{x}_i$. The vector $\mathbf{x}_i$ is being called feature vector or image[3].

The subject of cluster analysis is constituted, therefore, not so much by the original set of objects $\mathfrak{X}$, as by its representation, given through the matrix $X = (\mathbf{x}_1 \dots \mathbf{x}_m)^{\mathsf{T}}$, whose i-th row is a vector of features, describing the i-th object. In view of the fact that to object $\mathfrak{x}_i$ corresponds the i-th row of matrix X, the term ,,object" shall be used to denote both the element $\mathfrak{x}_i \in \mathfrak{X}$ and the vector of feature values $\mathbf{x}_i$, characterising this object. In statistics, vector $\mathbf{x}_i$ is called (n-dimensional) observation. Even though the values of individual measurements of feature values might be expressed in different scales (nominal, ordinal or quotient), the majority of practical and theoretical results have been obtained under the assumption that the components of vectors $\mathbf{x}_i$ are real numbers. Thus, for a vast majority of cases, considered in the present book, we shall be assuming that the observations are given by the vectors $\mathbf{x}_i \in \mathbb{R}^n$.

It is alternatively assumed that information on the data set is provided in the form of the matrix S of dimension $m \times m$, the elements of this matrix s_{ij} represent similarities (or dissimilarities) for the pairs of objects $\mathfrak{x}_i$, $\mathfrak{x}_j$. When making use of such a representation, we can give up the requirement of having the features, describing the objects, measured on the quantitative scales. The pioneer of such a perspective was Polish anthropologist, ethnographer, demographer and statistician – Jan Czekanowski[4]. The method, developed by Czekanowski, and presented in the first Polish handbook of modern methods of data analysis and interpretation of its results [86], consists in replacing numbers in the matrix S by the appropriately selected graphical symbols. In this manner, an unordered diagram arises, which, after an adequate reordering of rows and columns of the matrix, makes apparent the existence of groups of objects mutually similar. Although the method was developed originally mora than 100 years ago, it is still in use, mainly in archeology[5], in economic sciences[6], and even in musicology.

---

[3] The latter term is particularly justified, when we treat the measurements as mappings $\mathbf{f}\colon \mathfrak{X} \to \mathbb{R}^n$ of the set of objects into a certain set of values. Then, $\mathbf{x}_i = \mathbf{f}(\mathfrak{x}_i)$, and in the mathematical nomenclature $\mathbf{x}_i$ is the image of the object $\mathfrak{x}_i$.

[4] See J.Gajek. Jan Czekanowski. Sylwetka uczonego. *Nauka Polska*, 6(2), 1958, 118-127.

[5] See, e.g., A. Soltysiak, and P. Jaskulski. Czekanowski's Diagram: A method of multidimensional clustering. In: J.A. Barceló, I. Briz and A. Vila (eds.) *New Techniques for Old Times. CAA98. Computer Applications and Quantitative Methods in Archaeology.* Proceedings of the 26th Conf., Barcelona, March 1998 (BAR International Series 757). Archaeopress, Oxford 1999, pp. 175-184.

[6] See, e.g., A. Wójcik. Zastosowanie diagramu Czekanowskiego do badania podobieństwa krajów Unii Europejskiej pod względem pozyskiwania energii ze źródeł odnawialnych. *Zarządzanie i Finanse* (*J. of Management and Finance*), 11(4/4), 353-365, 2013, `http://zif.wzr.pl/pim/2013_4_4_25.pdf`

Using the method of Czekanowski, the mathematicians from Wrocław: K. Florek, J. Łukaszewicz, J. Perkal, H. Steinhaus and S. Zubrzycki, elaborated the so-called ,,Wrocław taxonomy", which they presented in 1957 in 17-th issue of *Przegląd Antropologiczny.* In further parts of the book we present other methods of analysing the matrix of similarities. More advanced considerations of application of the similarity matrix in cluster analysis have been presented in the references [32] and [33].

The role of the classical cluster analysis is to split the set of objects (observations) into $k < m$ groups $\mathcal{C} = \{C_1, \dots, C_k\}$, where each i-th group C_i is called cluster. Such a division fulfils three natural requirements:

(i) Each cluster ought to contain at least one object, $C_j \neq \emptyset, j = 1, \dots, k$.
(ii) Each object ought to belong to a certain cluster, $\bigcup_{j=1}^{k} C_j = \mathfrak{X}$.
(iii) Each object ought to belong to exactly one cluster, $C_{j_1} \cap C_{j_2} = \emptyset, j_1 \neq j_2$.

In particular, when $k = m$, each cluster contains exactly one element from the set $\mathfrak{X}$. This partition is trivial, and so we shall be considering the cases, in which k is much smaller than m.

An exemplary illustration of the problem that we deal with in cluster analysis, is provided in the Figure 2.1. The "clouds" of objects, situated in the lower left and upper right corners constitute distinctly separated clusters. The remaining objects form three, two, or one cluster, depending on how we define the notion of similarity between the objects.

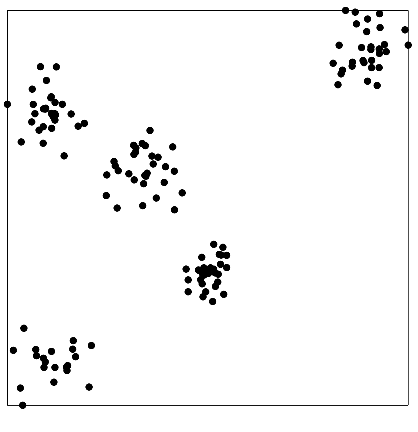


Fig. 2.1. The task of cluster analysis: to break down the set of observations into k disjoint subsets, composed of similar elements.

Cluster analysis is also referred to as unsupervised learning. We lack here, namely, the information on the membership of the objects in classes, and it is not known, how many classes there should really be. Even though the mechanical application of the algorithms, which are presented in the further parts of the book allows for the division of any arbitrary set into a given number of classes, the partition thus obtained may not have any sense. Assume that $\mathfrak{X}$ is a set of points selected conform to the uniform distribution from the set $[0,1] \times [0,1]$ – see Figure 2.2(a). Mechanical application of an algorithm of grouping, with the predefined parameter $k = 3$ leads to the result, shown in Figure 2.2b. It is obvious that although the partition obtained fulfills the conditions set before, it has no sense.

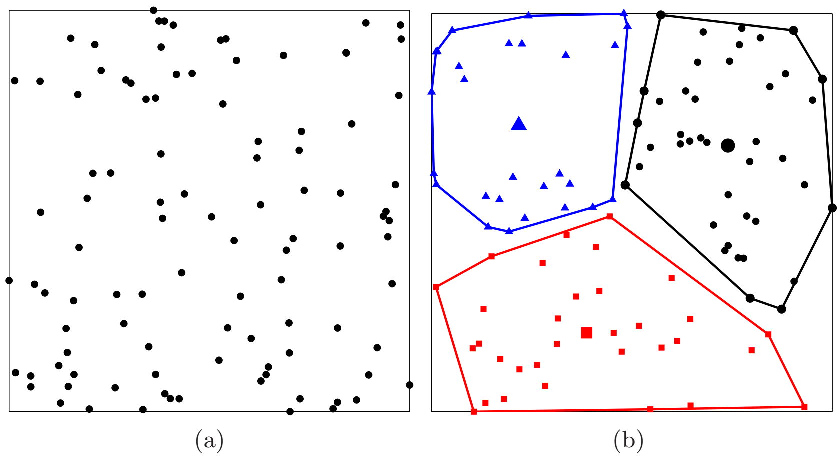


Fig. 2.2. Grouping of the set of randomly generated objects. (a): 100 points randomly selected from the set $[0,1] \times [0,1]$. (b): Division of the objects into three groups using the k-means algorithm. Bigger marks indicate the geometrical centers of groups.

In informal terms, the presence of a structure in a data set is manifested through the existence of separate areas, and hence of clusters, enjoying such a property that any two objects, belonging to the common cluster C_i are more mutually similar than any two objects, picked from two different clusters, i.e.

$$s(\mathfrak{x}', \mathfrak{x}'') > s(\mathfrak{y}', \mathfrak{y}'')$$

if only $\mathfrak{x}', \mathfrak{x}'' \in C_i$, $\mathfrak{y}' \in C_{j_1}$, $\mathfrak{y}'' \in C_{j_2}$ and $j_1 \neq j_2$. Symbol s denotes here a certain measure of similarity, that is, a mapping $s\colon \mathfrak{X} \times \mathfrak{X} \to \mathbb{R}$. In many situations it is more convenient to make use of the notion of dissimilarity (like, e.g., distance) and require the objects, belonging to different clusters, to be more distant than the objects, belonging to the same cluster. Various definitions of the measures of similarity or dissimilarity are considered in the subsequent chapter. The choice of the appropriate measure constitutes an additional factor of complexity of the data analysis task. Yet another such factor is the choice of an adequate criterion for determining the partition of the set $\mathfrak{X}$.

The majority of the methods of grouping consists in an "intelligent" extraction of information from the matrix S, with elements representing similarity or dissimilarity of the pairs of objects. An excellent example of this kind of procedure is provided by the hierarchical methods, shortly presented in Section 2.3 or by the spectral methods, being the primary subject of Chapter 5 .

The most popular methods of grouping include the hierarchical methods, the combinatorial methods (referred also to as relocation-based), the density-based methods, the grid methods and the methods based on models. The descriptions of these methods and comments on them can be found in numerous survey studies, like, e.g. [192], [190], or [373]. In the further part of the chapter we shall present their short characteristics.

2.2 Measures of similarity/dissimilarity

In order to be able to quantify asssociations between pairs of objects, a measure of similarity $s\colon \mathfrak{X} \times \mathfrak{X} \to \mathbb{R}$ or of dissimilarity is introduced. The two measures are, in principle, dual, that is - the lower the value of dissimilarity, the more similar the two compared objects. A particular example of dissimilarity is distance (metric), that is, a function $d\colon \mathfrak{X} \times \mathfrak{X} \to \mathbb{R}_+ \cup \{0\}$ fulfilling three conditions:

(a) $d(\mathfrak{x}, \mathfrak{y}) = 0$ if and only if $\mathfrak{x} \equiv \mathfrak{y}$,
(b) $d(\mathfrak{x}, \mathfrak{y}) = d(\mathfrak{y}, \mathfrak{x})$ (symmetry),
(c) $d(\mathfrak{x}, \mathfrak{y}) \le d(\mathfrak{x}, \mathfrak{z}) + d(\mathfrak{z}, \mathfrak{y})$ (triangle inequality),

for arbitrary $\mathfrak{x}, \mathfrak{y}, \mathfrak{z} \in \mathfrak{X}$. When only conditions (b) and (c) are satisfied, then d is called pseudo-distance.

For instance, if d_{max} denotes the maximum value of distance between the pairs of objects from the set $\mathfrak{X}$, then distance can be transformed into a measure of similarity (proximity) $s(\mathfrak{x}_i, \mathfrak{x}_j) = d_{max} - d(\mathfrak{x}_i, \mathfrak{x}_j)$. The thus obtained measure of proximity attains the maximum values, when $i = j$ (object $\mathfrak{x}_i$ is identical with itself), and the lower the value of this measure, the less mutually similar (more dissimilar) the objects compared are.

In the above example, the maximum value of similarity is the number $d_{max} = s(\mathfrak{x}_i, \mathfrak{x}_i)$. It is more convenient to operate with the normalised similarity $s(\mathfrak{x}_i, \mathfrak{x}_j) = 1 - d(\mathfrak{x}_i, \mathfrak{x}_j)/d_{max}$. Generally, if $f\colon \mathbb{R} \to \mathbb{R}$ is a monotonously decreasing function, such that $f(0) > 0$ and $\lim_{\xi \to \infty} f(\xi) = a \ge 0$, then

$$s(\mathfrak{x}_i, \mathfrak{x}_j) = f(d(\mathfrak{x}_i, \mathfrak{x}_j)) \tag{2.1}$$

is a measure of similarity, induced by the distance d. Another, frequently applied example of ther measure of similarity is provided by the transformation $s(\mathfrak{x}_i, \mathfrak{x}_j) = \exp[-d^2(\mathfrak{x}_i, \mathfrak{x}_j)/\sigma^2]$, where $\sigma > 0$ is a parameter, or, yet, $s(\mathfrak{x}_i, \mathfrak{x}_j) = 1/[d(\mathfrak{x}_i, \mathfrak{x}_j) + \epsilon]$, where $\epsilon > 0$ is a small number[7]. The condition, given above, $a \ge 0$ is, in principle, not necessary; it is introduced in order to preserve the symmetry with the non-negative values of the distance measure d.

[7] In practice, a small number means one that compared to distances it is small but still numerically significant under the available machine precision.

Note also that if d is a distance, then the measure of similarity, defined through the transformation f, reffered to above, fulfills the triangle condition in the form - see [72]:

$$s(\mathfrak{x}, \mathfrak{y}) + s(\mathfrak{y}, \mathfrak{z}) \leq s(\mathfrak{x}, \mathfrak{z}) + s(\mathfrak{y}, \mathfrak{y})$$

Since the measures of similarity and dissimilarity are dual notions, see, e.g., [72], we shall be dealing in further course primarily with various measures of dissimilarity, and in particular – with distances. Making use of distances requires having the possibility of assigning to each object $\mathfrak{x}_i$ its representation $\mathbf{x}_i$. An interesting and exhaustive survey of various measures of similarity/dissimilarity can be found, for instance, e.g., in [66].

2.2.1 Comparing the objects having quantitative features

When all the features, which are used to describe objects from the set $\mathfrak{X}$ are quantitative, then every object $\mathfrak{x}_i \in \mathfrak{X}$ is identified with an n-dimensional vector $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{in})^{\mathrm{T}}$. The most popular measure of dissimilarity is the Euclidean distance

$$d(\mathfrak{x}_i, \mathfrak{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\| = \sqrt{\sum_{l=1}^{n} (x_{il} - x_{jl})^2}$$

or, more generally, the norm defined by the square form

$$d_W(\mathfrak{x}_i, \mathfrak{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|_W = \sqrt{(\mathbf{x}_i - \mathbf{x}_j)^{\mathrm{T}} W (\mathbf{x}_i - \mathbf{x}_j)} \tag{2.2}$$

where W is a positive definite matrix of the dimensions $n \times n$.

If W is a unit matrix, then equation (2.2) defines the Euclidean distance. If, on the other hand, W is a diagonal matrix having the elements

$$w_{ij} = \begin{cases} \omega_i & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

then (2.2) defines the weighted Euclidean distance, i.e.

$$d_W(\mathfrak{x}_i, \mathfrak{x}_j) = \sqrt{\sum_{l=1}^{n} \omega_l (x_{il} - x_{jl})^2} = \sqrt{\sum_{l=1}^{n} (y_{il} - y_{jl})^2}$$

where $y_{il} = \sqrt{\omega_l} x_{il}$ is the weighted value of the feature l, measured for the i-th object.

The Euclidean distance is being generalized in various manners. These most commonly used are commented upon below.

2.2.1.1 Minkowski distance

Minkowski distance (norm) is defined as follows

$$d_p(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|_p = \Big[\sum_{l=1}^{n} |x_{il} - x_{jl}|^p\Big]^{1/p}, \ p \geq 1, p \in \mathbb{R} \tag{2.3}$$

When we take $p = 1$, we obtain the city block distance (called also taxicab or Manhattan distance)

$$d_1(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|_1 = \sum_{l=1}^{n} |x_{il} - x_{jl}| \tag{2.4}$$

For $p = 2$, equation (2.3) defines the Euclidean distance. In view of the popularity of this distance definition, we shall be writing $\|\mathbf{x}_i - \mathbf{x}_j\|$ instead of $\|\mathbf{x}_i - \mathbf{x}_j\|_2$.

Finally, when $p = \infty$, we get Chebyshev distance

$$d_\infty(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|_\infty = \max_{l=1,\ldots,n} |x_{il} - x_{jl}| \tag{2.5}$$

Minkowski distance is used not only in exact sciences, but also in psychology[8], industrial design and generally in designing. Unit circles are described in Minkowski metric by the equation

$$|x|^p + |y|^p = 1$$

which is also called the curve (or oval) of Lamé. The respective shapes for three values of the parameter p are presented in Figure 2.3a. Danish mathematician Piet Hein[9] concluded that the case of $p = 2.5$ leads to the shape, featuring *high aesthetic qualities*, see Figure 2.3b, this fact having been made use of in designing Sergels roundabout in Stockholm.

Distances, deriving from the Minkowski metric, have two important shortcomings. First, as the dimensionality of the problem increases, the difference between the close and the far points in the space $\mathbb{R}^n$ disappears, this being the effect of summing of the differences in the locations of objects in the particular dimensions[10].

More precisely, the situation is as follows. Denote by $d_{p,n}^{min}, d_{p,n}^{max}$, respectively, the minimum and the maximum values of distance (measured with distance d_p) between two arbitrary points, selected from the set of m randomly generated points in n-dimensional space. Then, see [4]

[8] See chapter 3 in: C.H. Coombs, R.M. Dawes, A. Tversky. *Mathematical Psychology: An Elementary Introduction*. Prentice Hall, Englewood Cliffs, NJ 1970,

[9] His profile can be found on the website `http://www.piethein.com/`.

[10] Equivalently, one can say that two arbitrary vectors in $\mathbb{R}^n$ are orthogonal [289, p. 7.1.3].

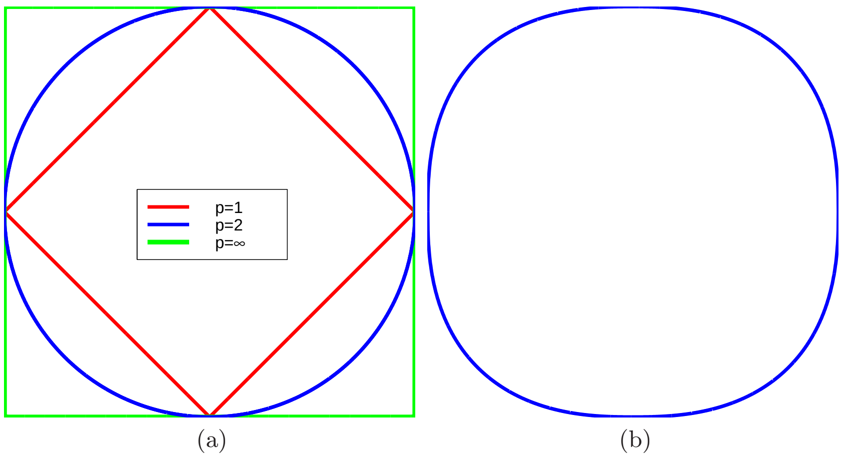


Fig. 2.3. Unit circles for the selected values of the parameter p: (a) $p = 1, 2, \infty$, (b) $p = 2.5$ (Hein's super-ellipse)

$$C_p \leq \lim_{n\to\infty} \mathbb{E}\Big[\frac{d_{p,n}^{max} - d_{p,n}^{min}}{n^{1/p-1/2}}\Big] \leq (m-1)C_p \tag{2.6}$$

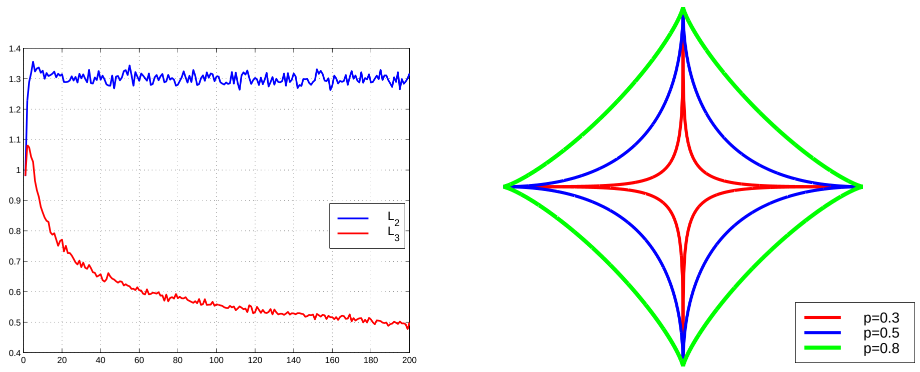
where C_p is a constant, depending upon the value of p, and $\mathbb{E}$ denotes the expected value. This inequality implies that in a high-dimensional space the difference $d_{p,n}^{max} - d_{p,n}^{min}$ increases proportionally to $n^{1/p-1/2}$ irrespective of the distribution of data [4]. This property plays a dominating role, when $n \geq 15$. In particular (see Figure 2.4a)

$$d_{p,n}^{max} - d_{p,n}^{min} \to \begin{cases} C_1\sqrt{n} & \text{if } p = 1 \\ C_2 & \text{if } p = 2 \\ 0 & \text{if } p \geq 3 \end{cases}$$

In order to prevent this phenomenon, Aggarwal, Hinnenburg and Keim proposed in [4] application of the fractional Minkowski distances with the parameter $p \in (0, 1]$ – see Figure 2.4(b). Yet, in this case (2.3) is no longer a distance, since the triangle condition is not satisfied. If, for instance, $\mathbf{x} = (0, 0)$, $\mathbf{y} = (1, 1)$ and $\mathbf{z} = (1, 0)$, then

$$d(\mathbf{x}, \mathbf{y}) = 2^{1/p} > d(\mathbf{x}, \mathbf{z}) + d(\mathbf{y}, \mathbf{z}) = 1 + 1$$

A subsequent, but not so critical issue is constituted by the fact that the values of the Minkowski metric are dominated by these features, whose values are measured on the scales with the biggest ranges. This issue can be relatively easily resolved by introducing weighted distance, that is - by replacing each component of the equation (2.3) by the expression $\omega_l(x_{il} - x_{jl})^p$, where w_l is the weight equal, e.g., the inverse of the standard deviation of the l-th feature, or the inverse of the range of variability of the l-th feature. The counterpart to the second variant is constituted by the initial normalization of data, ensuring



(a) (b)

Fig. 2.4. The influence of the parameter p on the properties of Minkowski distance, (a): Average values of the difference between the most distant points from a 100-point set in dependence upon the number of dimensions, n, and the value of the exponent, p, (b): Unit circles for $p < 1$

that $x_{il} \in [0,1]$ for each of the features $l = 1,\dots,n$. This is, usually, a routine procedure, preceding the proper data analysis. In some cases, instead of a simple normalization, standardization is applied, that is - the original value x_{il} is replaced by the quotient $(x_{il} - \mu_l)/\sigma_l$, where μ_l, σ_l are the average value and the standard deviation of the l-th feature.

2.2.1.2 Mahalanobis distance

When defining distance (2.3) it is by default assumed that features are not mutually correlated. When this assumption is not satisfied, Mahalanobis distance is usually applied,

$$d_{\Sigma}(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{(\mathbf{x}_i - \mathbf{x}_j)^{\mathrm{T}} \Sigma^{-1} (\mathbf{x}_i - \mathbf{x}_j)} \tag{2.7}$$

which is a variant of the distance (2.2), where W is equal the inverse of the covariance matrix. Covariance matrix is calculated in the following manner

$$\boldsymbol{\Sigma} = \frac{1}{m} \sum_{i=1}^{m} (\mathbf{x}_i - \overline{\boldsymbol{\mu}})(\mathbf{x}_i - \overline{\boldsymbol{\mu}})^{\mathrm{T}} \tag{2.8}$$

with

$$\overline{\boldsymbol{\mu}} = \frac{1}{m} \sum_{i=1}^{m} \mathbf{x}_i \tag{2.9}$$

being the vector of average values.

Note that:

(a) By applying the transformation $\mathbf{y}_i = \Sigma^{-1/2}\mathbf{x}_i$ we reduce Mahalanobis distance $d_\Sigma(\mathbf{x}_i, \mathbf{x}_j)$ to Euclidean distance between the transformed vactors, that is $d_\Sigma(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{y}_i - \mathbf{y}_j\|$.
(b) When the features are independent, then the covariance matrix is a diagonal matrix: the nonzero elements are equal the variances of the particular features. In such a case Mahalanobis distance becomes the weighted Euclidean distance of the form

$$d_S(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{\sum_{l=1}^{n} \left(\frac{x_{il} - x_{jl}}{s_l}\right)^2} \tag{2.10}$$

Mahalanobis distance is useful in identification of the *outliers* (atypical observations). A number of properties of this distance are provided in the tutorial [242].

2.2.1.3 Bregman divergence

The distances, that is – the measures of dissimilarity – considered up till now, have a differential character: $d(\mathbf{x}, \mathbf{y}) = \varphi(\mathbf{x} - \mathbf{y})$, where $\varphi\colon \mathbb{R}^n \to \mathbb{R}$ is an appropriately selected function. In certain situations, e.g. in problems concerning signal compression, measures are needed that would account for more complex relations between the vectors compared. An exhaustive survey thereof is given in [42]. An instance is represented in this context by the measure, introduced for the systems of speech compression by Chaffee[11]

$$d_{Ch}(\mathbf{x}, \mathbf{y}) = (\mathbf{x} - \mathbf{y})^{\mathrm{T}} R(\mathbf{x})(\mathbf{x} - \mathbf{y}) \tag{2.11}$$

While in the case of Mahalanobis distance (2.7) the covariance matrix, which appears there, is established a priori, the matrix of weights, which is used in the definition (2.11) depends upon the currently considered object $\mathbf{x}$. Another example is provided by the measure of Itakura-Saito[12].

All these measures are generalized by the so-called Bregman divergence. It is defined as follows, [37]

Definition 2.2.1 *Let $\phi\colon S \to \mathbb{R}$ be a strictly convex function, defined on a convex set $S \subset \mathbb{R}^n$. Besides, we assume that the relative interior, $rint(S)$, of the set S is not empty, and ϕ is a function differentiable on $rint(S)$. Bregman divergence is then such a function $d_\phi\colon S \times rint(S) \to [0, \infty)$ that for the arbitrary* $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$

[11] D.L. Chaffee, Applications of rate distortion theory to the bandwidth compression of speech, Ph.D. dissertation, Univ. California, Los Angeles, 1975. See also R.M. Gray, *et al.*, Distortion measures for speech processing, *IEEE Trans. on Acoustics, Speech and Signal Processing*, **28**(4), 367-376, Aug. 1980.

[12] See F. Itakura, S. Saito, Analysis synthesis telephony based upon maximum likelihood method, *Repts. of the 6th Intl. Cong. Acoust.* Tokyo, C-5-5, C-17-20, 1968.

$$d_\phi(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x}) - \phi(\mathbf{y}) - (\mathbf{x}-\mathbf{y})^T \nabla\phi(\mathbf{y}) \tag{2.12}$$

Symbol $\nabla\phi(\mathbf{y})$ *denotes the gradient of the function* $\phi(\mathbf{y})$*.* □

Examples of Bregman divergence are shown in Table 2.1. Special attention ought to be paid to the last three examples. If we take for $\phi(\mathbf{x})$ the squared length of vector $\mathbf{x}$, then $d_\phi(\mathbf{x}, \mathbf{y})$ is equal squared Euclidean distance between the points, represented by the vectors $\mathbf{x}$ and $\mathbf{y}$. If the mapping ϕ is defined as $\mathbf{x}^{\mathrm{T}} W \mathbf{x}$, where W is a positive definite matrix, then we obtain the squared distance (2.2), in particular – defined in Subsection 2.2.1.2 Mahalanobis distance. Finally, if $\mathbf{x}$ is a stochastic vector[13], while ϕ is a negative value of entropy, $\phi(\mathbf{x}) = \sum_{j=1}^n x_j \log_2 x_j$, then we obtain Kullback-Leibler divergence, referred to frequently as KL-divergence. This notion plays an important role in information theory, machine learning, and in information retrieval.

Domain	$\phi(\mathbf{x})$	$d_\phi(\mathbf{x}, \mathbf{y})$	Divergence
$\mathbb{R}$	$\mathbf{x}^2$	$(\mathbf{x}-\mathbf{y})^2$	quadratic loss function
$\mathbb{R}_+$	$\mathbf{x}\log\mathbf{x}$	$\mathbf{x}\log(\frac{\mathbf{x}}{\mathbf{y}}) - (\mathbf{x}-\mathbf{y})$	
[0,1]	$\mathbf{x}\log\mathbf{x} + (1-\mathbf{x})\log(1-\mathbf{x})$	$\mathbf{x}\log(\frac{\mathbf{x}}{\mathbf{y}}) + (1-\mathbf{x})\log(\frac{1-\mathbf{x}}{1-\mathbf{y}})$	logistic loss function
$\mathbb{R}_{++}$	$-\log\mathbf{x}$	$\frac{\mathbf{x}}{\mathbf{y}} - \log(\frac{\mathbf{x}}{\mathbf{y}}) - 1$	Itakura-Saito distance
$\mathbb{R}^n$	$\lVert\mathbf{x}\rVert^2$	$\lVert\mathbf{x}-\mathbf{y}\rVert^2$	squared Euclidean distance
$\mathbb{R}^n$	$\mathbf{x}^{\mathrm{T}} W x$	$(\mathbf{x}-\mathbf{y})^{\mathrm{T}} W (\mathbf{x}-\mathbf{y})$	Mahalanobis distance
n-simplex	$\sum_{j=1}^n \mathbf{x}_j \log_2 \mathbf{x}_j$	$\sum_{j=1}^n \mathbf{x}_j \log_2(\frac{x_j}{y_j})$	KL-divergence

Table 2.1. Bregman divergences generated by various convex functions,[37]

2.2.1.4 Cosine distance

Another manner of coping with the "curse of dimensionality" is suggested by, for instance, Hamerly, [159], namely by introducing distance $d_{cos}(\mathbf{x}_i, \mathbf{x}_j)$ defines as 1 minus cosine of the angle between the vectors $\mathbf{x}_i, \mathbf{x}_j$,

$$d_{cos}(\mathbf{x}_i, \mathbf{x}_j) = 1 - \cos(\mathbf{x}_i, \mathbf{x}_j) = 1 - \frac{\mathbf{x}_i^{\mathrm{T}} \mathbf{x}_j}{\lVert\mathbf{x}_i\rVert \lVert\mathbf{x}_j\rVert} = 1 - \frac{\sum_{l=1}^n x_{il} x_{jl}}{\lVert\mathbf{x}_i\rVert \lVert\mathbf{x}_j\rVert} \tag{2.13}$$

The value of cosine of an angle, appearing in the above formula, constitutes and example of a measure of similarity, which is the basic measure, applied in the information retrieval systems for measuring the similarity between the documents [29]. In this context the components $\mathbf{x}_{il}$ represent the frequency

[13] That is – all of its components are non-negative and $\sum_{j=1}^n x_j = 1$.

of appearance of a keyword indexed l in the i-th document. Since frequencies are non-negative, then, for two arbitrary vectors, representing documents, $0 \le \cos(\mathbf{x}_i, \mathbf{x}_j) \le 1$ and $0 \le d_{cos}(\mathbf{x}_i, \mathbf{x}_j) \le 1$. Other measures, which quantify the similarity of documents, are considered in [181].

2.2.1.5 Power distance

When we wish to increase or decrease the growing weight, assigned to these dimensions, for which the objects considered differ very much, we can apply the so-called power distance[14]:

$$d_{p,r}(\mathbf{x}_i, \mathbf{x}_j) = \Big(\sum_{l=1}^{n} |x_{il} - x_{jl}|^p \Big)^{1/r} \tag{2.14}$$

where p and r are parameters. The parameter p controls the increasing weight, which is assigned to the differences for the particular dimensions, while parameter r controls the increasing weight, which is assigned to the bigger differences between objects. Of course, when $p = r$, this distance is equivalent to Minkowski distance.

2.2.2 Comparing the objects having qualitative features

Similarly as in the preceding point, we assume here, that for reasons related to the facility of processing, each object is represented by the vector $\mathbf{x}$, but now its components are interpreted more like labels. If, for instance, the feature of interest for us is *eye color*, then this feature may take on such values as 1 – blue, 2 – green, etc. It is essential that for such labels there may not exist a natural order of such label "values", proper for a given phenomenon under consideration. In such situations one can use as the measure of dissimilarity the generalized Hamming distance: $d_H(\mathbf{x}_i, \mathbf{x}_j)$, equal the number of these features, whose values for the compared objects are different.

When we deal with mixed data, that is – a part of features have a qualitative character, and a part – quantitative character, then we can apply the so-called Gower coefficient [213], which is the weighted sum of the partial coefficients of divergence $\delta(i, j, l)$, determined for each feature of the objects $\mathbf{x}_i$ i $\mathbf{x}_j$. For a nominal (qualitative) feature we take $\delta(i, j, l) = 1$, when the value of this feature in both objects is different and $\delta(i, j, l) = 0$ in the opposite case. If, on the other hand, we deal with the quantitative feature having values from the interval $[x_l^{min}, x_l^{max}]$, then we replace $\delta(i, j, l)$ by

$$\delta(i, j, l) = \frac{|x_{il} - x_{jl}|}{x_l^{max} - x_l^{min}}$$

Ultimately, we take as the distance between two objects

[14] Por. np. *Web-based handbook of statistics. Cluster analysis: Agglomeration.* `http://www.statsoft.pl/textbook/stathome.html`.

$$d_m(\mathbf{x}_i, \mathbf{x}_j) = \frac{\sum_{l=1}^{n} w(i,j,l)\delta(i,j,l)}{\sum_{l=1}^{n} w(i,j,l)} \tag{2.15}$$

where $w(i,j,l)$ is the weight equal zero when either one of the values x_{il}, x_{jl} was not observed, or we deal with a so-called asymmetric binary feature[15] and for one of the compared objects the value of this feature is equal zero. In the remaining cases we have $w(i,j,l) = 1$.

The recommender systems [60], whose purpose is to suggest to the users the selection of broadly understood goods (books, movies, discs, information, etc.) matching in a possibly best manner the tastes and the preferences of the users, make use of the similarity measure $s(\mathbf{x}_i, \mathbf{x}_j)$ between the preferences of the given user, $\mathbf{x}_i$, and the preferences of other users, $\mathbf{x}_j$, where $j = 1, \ldots, m, j \neq i$. Here, the l-th component of the preference vector corresponds to the evaluation of the l-th good. One of the most often applied measures of similarity is, in this case, the modified Pearson correlation coefficient

$$r(\mathbf{x}_i, \mathbf{x}_j) = \frac{(\mathbf{x}_i - \overline{\mathbf{x}}_i) \cdot (\mathbf{x}_j - \overline{\mathbf{x}}_j)}{\|\mathbf{x}_i - \overline{\mathbf{x}}_i\| \|\mathbf{x}_j - \overline{\mathbf{x}}_j\|} \tag{2.16}$$

where $\overline{\mathbf{x}}_i$ denotes the average of the vector $\mathbf{x}_i$. Modification concerns the numerator of the above expression: when summing up the respective products one accounts for only those components of the vectors, which represent the common evaluations of the users compared. Instead of Pearson correlation one can apply, of course, other measures of correlation, adapted to the character of the features used in describing objects. The most popular variants applied for the qualitative features are Spearman or Kendall correlations.

Just like in the case of the cosine similarity measure, also here one can introduce the correlation-based distance $d_{r1}(\mathbf{x}_i, \mathbf{x}_j) = 1 - r(\mathbf{x}_i, \mathbf{x}_j)$, taking values from the interval $[0, 2]$. Another variant was proposed in [339]: $d_{r2}(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{2[1 - r(\mathbf{x}_i, \mathbf{x}_j)]}$. This distance also takes values from the interval $[0, 2]$; the positively and strongly correlated variables correspond to small distances, while negatively and strongly correlated variables correspond to large distances, the weakly correlated variables being situated midway. One should also note that when we deal with the centered variables (i.e. $\overline{\mathbf{x}}_i = \overline{\mathbf{x}}_j = 0$), then the value of the correlation coefficient is identical with the value of cosine of the angle between the two vectors. This fact was taken advantage of by Trosset [339] to formulate the angular representation of correlation, used then to construct a new algorithm of cluster analysis. Finally, in [142], side by side with d_{r2}, the (pseudo-)distance

$$d_{r3}(\mathbf{x}_i, \mathbf{x}_j) = \Big(\frac{1 - r(\mathbf{x}_i, \mathbf{x}_j)}{1 + r(\mathbf{x}_i, \mathbf{x}_j)}\Big)^{\beta}$$

where $\beta > 0$ is a parameter, was introduced. This distance was applied in the FCM algorithm (which is presented in Section 3.3). The coefficient β controls, in

[15] E.g. in medical tests it is often assumed that lack of a given feature for a patient is denoted by symbol 0, while its presence – by the symbol 1. In such situations it is better not to account in the comparisons for the number of zeroes.

this case, the degree of fuzziness of the resulting partition. When $r(\mathbf{x}_i, \mathbf{x}_j) = -1$, then the distance is undefined.

Since the value of r represents the cosine of the angle between the (centered) vectors $\mathbf{x}_i$, $\mathbf{x}_j$, then

$$d_{\tan}(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{\frac{1 - r^2(\mathbf{x}_i, \mathbf{x}_j)}{r^2(\mathbf{x}_i, \mathbf{x}_j)}} \tag{2.17}$$

can be treated as tangent distance. The tangent distance measure, $d_{\tan}$, is based on the volume of joint information, carried by the features analysed. The parallel vectors ($r = 1$), as well as the anti-parallel ones ($r = -1$) carry the very same information, and hence their distance is equal 0, while similarity is the highest and equal 1. On the other hand, the orthogonal vectors are infinitely distant and have zero similarity, since each of them carries entirely different information. Vectors, having correlation coefficients different from 0 and ± 1 contain partly a specific information, and partly common information. The volume of the common information constitutes the measure of their similarity. The tangent measure of distance has nowadays a wide application in the analysis of similarity[16]. Another, normalized variant of the correlation-based similarity measure is

$$r_n(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{2}\Big(r(\mathbf{x}_i, \mathbf{x}_j) + 1\Big) \tag{2.18}$$

which guarantees that $r(\mathbf{x}_i, \mathbf{x}_j) \in [0, 1]$. In bioinformatics the s-called squared correlation distance is being applied $d_b(\mathbf{x}_i, \mathbf{x}_j) = 1 - r^2(\mathbf{x}_i, r_j)$ when comparing genetic profiles [17]. One can find a number of interesting comments on the properties of the correlation coefficient in [294].

Pearson correlation is an adequate yardstick when the variables compared are normally distributed. When this is not the case, other measures of similarity ought to be applied.

Let us also mention the chi-square distance, which is defined as follows

$$d_{\chi^2}(\mathbf{x}, \mathbf{y}) = \frac{1}{2}\sum_{i=1}^{n} \frac{(x_i - y_i)^2}{x_i + y_i} \tag{2.19}$$

and which is used in comparing histograms[18]. This distance finds application in correspondence analysis, as well as in the analysis of textures of digital images.

[16] see, e.g., J. Mazerski. *Podstawy chemometrii*, Gdańsk 2004. Electronic edition available at `http://www.pg.gda.pl/chem/Katedry/Leki_Biochemia/dydaktyka/chemometria/podstawy_chemometrii.zip`.

[17] See `http://www.improvedoutcomes.com/docs/WebSiteDocs/Clustering/Clustering_Parameters/Pearson_Correlation_and_Pearson_Squared_Distance_Metric.htm`.

[18] See V. Asha, N.U. Bhajantri, and P. Nagabhushan: GLCM-based chi-square histogram distance for automatic detection of defects on patterned textures. *Int. J. of Computational Vision and Robotics*, **2**(4), 302-313, 2011

Another measure, which is used in this context, is the Bhattacharyya distance, which measures the separability of classes; this distance is defined as follows:

$$d_B(\mathbf{x}, \mathbf{y}) = \Big(1 - BC(\mathbf{x}, \mathbf{y})\Big)^{1/2} \tag{2.20}$$

where $BC(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} \sqrt{x_i y_i}$ is the so-called Bhattacharyya coefficient. Sometimes, the following definition is used: $d_B(\mathbf{x}, \mathbf{y}) = -\ln BC(\mathbf{x}, \mathbf{y})$.

A survey on the measures of similarity / dissimilarity, used in grouping of the time series is provided, for instance, in [232].

2.3 Hierarchical methods of cluster analysis

Hierarchical methods are among the traditional techniques of cluster analysis. They consist in successive aggregation or division of the observations and their subsets. Resulting from this kind of procedure there is a tree-like structure, which is referred to as dendrogram.

The agglomerative techniques start from the set of observations, each of which is treated as a separate cluster. Clusters are aggregated in accordance with the decreasing degree of similarity (or the increasing degree of dissimilarity) until one, single cluster is established. The manner of proceeding is represented by the following pseudo-code 2.1:

Algorithm 2.1 Algorithm of agglomerative cluster analysis

```
1: Initialization. Establish $m$ single-element clusters and calculate distance for each
   pair of such clusters. Memorize the distances calculated in the symmetric square
   matrix $D = [d_{ij}]$.
2: Find a pair $C_i$, $C_j$ of the clusters that are the closest to each other.
3: Form a new cluster $C_k = C_i \cup C_j$. In the generated dendrogram this corresponds
   to introducing a new node and connecting it with the nodes, corresponding to the
   clusters $C_i$, $C_j$.
4: Update the distance matrix, i.e. calculate the distance between the cluster $C_k$ and
   the remaining clusters, except for $C_i$ and $C_j$.
5: Remove from the matrix $D$ rows and columns, corresponding to the aggregated
   clusters $C_i$, $C_j$ and add a row and a column, for the new cluster $C_k$.
6: Repeat steps (2) - (5) until only one, single cluster is created.
```

In step 3 of the above algorithm we join together two closest clusters. By defining more precisely the notion, related to the new distances from the cluster thus created to the remaining ones, one obtains seven different variants of the agglomerative algorithms (abbreviations in brackets correspond to the names, introduced in [318]):

(a) Single linkage method or nearest neighbor method (*single linkage*): Distance between two clusters is equal to the distance between two closest elements belonging to different clusters. The resulting clusters form, in this case, long "chains". In order to find the optimum solution to the task, involving the method specified, the algorithms are used, referring to the minimum spanning tree [19].

(b) Complete linkage method or the farthest neighbor method (*complete linkage*): Distance between two clusters is equal to the distance between two farthest objects, belonging to different clusters. This method is most appropriate, when the real objects form well separated and compact clusters.
(c) Average linkage method (*unweighted pair-group average*, UPGA): Distance between two clusters is equal the average distance between all pairs of objects belonging to both clusters considered.
(d) Weighted pair-group average method (*weighted pair-group average*, WPGA): This method is similar to the preceding one, but calculations are carried out with the weights, equal the numbers of objects in the two clusters considered. This method is advised in cases, when we deal with clusters having distinctly different numbers of objects.
(e) Centroid method (*unweighted pair-group centroid*, UPGC): Distance between two clusters is equal to distance between their centroids (gravity centers).
(f) Method of weighted centroids or of the median (*weighted pair-group centroid*, WPGC): Distance between two clusters is calculated as in the previous method, but with introduction of weights, which are equal the numbers of objects in the clusters considered.
(g) Ward method of minimum variance. In this method the sum of squares of distances between objects and the center of the cluster, to which the objects belong, is minimized. This method, even though considered to be very effective, tends to form clusters having similar (low) cardinalities.

The opposition to the agglomerative techniques is constituted by the divisive techniques. Here, analysis starts from a single all-encompassing cluster, which is subject to successive divisions, according to the increasing degree of similarity. These techniques, even though apparently symmetric to the agglomerative ones, are used in practice much less frequently.

Since for a given set of observations one can obtain multiple different hierarchies, the question arises: "To what extent a dendrogram reflects the distances between the particular pairs from the set X?" One of the popular means for assessing the quality of grouping was introduced by Sokal and Rohlf[20], namely the *cophenetic correlation coefficient*. Given a dendrogram, the matrix D_T is formed, presenting the levels of aggregation, at which the pairs of objects appeared for the first time in the same cluster. Let further E be a vector (variable) formed of the elements located above the main diagonal of the distance matrix D and let T be the vector (variable) formed out of the elements situated above the main di-

[19] M. Delattre and P. Hansen. "Bicriterion cluster analysis", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol-2, No. 4, pp. 277-291, 1980

[20] R.R. Sokal, F.J. Rohlf. 1962. The comparison of dendrograms by objective methods. *Taxon*, 11(2), 1962,33-40

agonal[21] of the D_T matrix. The cophenetic correlation coefficient is the Pearson correlation coefficient between these two variables. The computational details are presented in the example 2.3.1. Another coefficient, which allows for assessing the degree of matching between the dendrogram and the matrix of distances (similarities) is the Goodman-Kruskal coefficient[22] (*Goodman-Kruskal gamma coefficient*, *gamma index*). It was introduced with the intention of assessing the concordance of orderings of features expressed on the ordinal scale.

The hierarchical methods have some important shortcomings. We mention below some of the most important ones:

(i) They lose their clarity with the increase of the number of analysed objects.
(ii) There is no way to shift objects from one cluster to another, even if they had been wrongly classified at the initial stages of the procedure.
(iii) The results reflect the degree, to which the data match the structure implied by the algorithm selected ("chain" or a compact "cloud").

Example 2.3.1 *Consider the data from the diagram 2.5(a). The matrix of distances between the objects is of the form*

$$D = \begin{array}{c|cccccc} & 1 & 2 & 3 & 4 & 5 & 6 \\ \hline 1 & 0 & 4.4721 & 4.2426 & 2.2361 & 2.8284 & 3.1623 \\ 2 & & 0 & 1.4142 & 3.0000 & 2.0000 & 3.1623 \\ 3 & & & 0 & 2.2361 & 1.4142 & 2.0000 \\ 4 & & & & 0 & 1.0000 & 1.0000 \\ 5 & & & & & 0 & 1.4142 \\ 6 & & & & & & 0 \end{array}$$

By applying the complete link (farthest neighbour) method, we construct the corresponding dendrogram. In the first step we aggregate the objects with numbers 4 and 6, between which distance is equal 1 unit. In the next step we add object number 5, situated at the distance $d(\{\mathfrak{x}_4, \mathfrak{x}_6\}, \mathfrak{x}_5) = \max\big(d(\mathfrak{x}_4, \mathfrak{x}_5), d(\mathfrak{x}_6, \mathfrak{x}_5)\big) = 1.4142$. *In the third step we glue together objects having numbers 2 and 3, between which distance is equal 1.4142. Then, in the fourth step, we aggregate the clusters formed until now into one cluster* $\{\{\{\mathfrak{x}_4, \mathfrak{x}_6\}, \mathfrak{x}_5\}, \{\mathfrak{x}_2, \mathfrak{x}_3\}\}$, *the distance between these clusters being equal* 3.1623. *Finally, in the last step, we add the element* $\mathfrak{x}_1$, *which is situated at the distance of 4.4721 from the cluster already established. The dendrogram obtained therefrom is shown in Figure 2.5(b). The matrix of the dendritic distances, determined on the basis of the calculations performed, is as follows:*

[21] Both distance matrices are symmetric, it suffices, therefore, to consider their elements situated above (or below) the main diagonal.

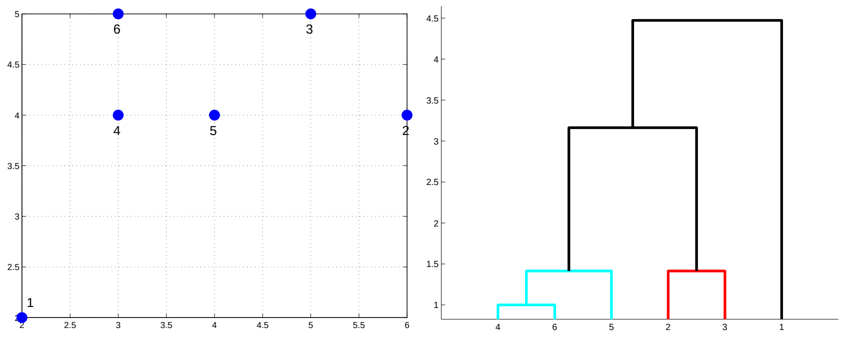
[22] L.A. Goodman, W.H. Kruskal. Measures of association for cross classifications. *J. of the American Statistical Association*, 49(268), 1954, 732-764

$$D_T = \begin{bmatrix} 0 & 4.4721 & 4.4721 & 4.4721 & 4.4721 & 4.4721 \\ & 0 & 1.4142 & 3.1623 & 3.1623 & 3.1623 \\ & & 0 & 3.1623 & 3.1623 & 3.1623 \\ & & & 0 & 1.4142 & 1.0000 \\ & & & & 0 & 1.4142 \end{bmatrix}$$

Hence, vectors E and T are equal

E	4.47	4.24	2.23	2.83	3.16	1.41	3.00	2.00	3.16	2.24	1.41	2.00	1.00	1.00	1.41
T	4.47	4.47	4.47	4.47	4.47	1.41	3.16	3.16	3.16	3.16	3.16	3.16	1.41	1.00	1.41

The cophenetic correlation coefficient, expressing the degree of match between the dendrogram from Figure 2.5(b) and the matrix of distances D is equal the coefficient of Pearson correlation $c(E, T) = 0.7977$. □



(a) (b)

Fig. 2.5. The exemplary data set (a) and the corresponding dendrogram, (b) obtained from the complete link method

2.4 Partitional clustering

Let $U = [u_{ij}]_{m \times k}$ denote the matrix with elements indicating the fact of assignment of i-th object to the j-th class. When $u_{ij} \in \{0, 1\}$, then we speak of a "crisp" partition of the set $\mathfrak{X}$, while for $u_{ij} \in [0, 1]$ – we deal with the "fuzzy" partition. The latter case is considered in Section 3.3. here, we concentrate on the "crisp" partitions.

If matrix U is supposed to represent the partition of the set of objects (see conditions, mentioned in Section 2.1), then this matrix has to satisfy the following conditions: (a) each object has to belong to exactly one cluster, that is $\sum_{j=1}^{k} u_{ij} = 1$ and (b) each cluster must contain at least one object, but cannot contain all the objects, i.e. $1 \leq \sum_{i=1}^{m} u_{ij} < m$. Denote by $\mathcal{U}_{m \times k}$ the set of all the matrices, representing the possible partitions of the set of m objects into k disjoint classes. It turns out, see, e.g., [10], [213], that for the given values of m and k the cardinality of the set $\mathcal{U}_{m \times k}$ is defined by the formula

$$\vartheta(m,k) = \frac{1}{k!} \sum_{j=1}^{k} (-1)^{k-j} \binom{k}{j} j^m \tag{2.21}$$

For instance, $\vartheta(3,2) = 3$, but already $\vartheta(100,5)$, is the number of the order of 10^{68}. Generally, the number of ways, in which m observations can be divided among k clusters is approximately equal $k^m/k!$, meaning that it is of the order of $O(k^m)$. In particular, when $k = 2$, then $\vartheta(m,2) = 2^{m-1} - 1$. The problem of selection of the proper partition is, therefore, an $\mathcal{NP}$-complete task of combinatorial optimisation.

An effective navigation over the sea of the admissible partitions is secured by the criteria of grouping and the methods of optimisation, coupled with them.

2.4.1 Criteria of grouping based on dissimilarity

The fundamental criteria, applied in the partitional clustering are homogeneity and separation. Homogeneity of a cluster means that two arbitrary objects, which belong to it, are sufficiently similar, while separation of clusters means that two arbitrary objects, belonging to different clusters are sufficiently different. In other words, the partition, induced by the clustering algorithm should contain homogeneous and well separated groups of objects.

Let $D = [d_{ij}]_{m \times m}$ denote the matrix with elements d_{ij} corresponding to the dissimilarities of objects i and j. We assume that D is a symmetric and non-negative matrix, having zeroes on the diagonal. The homogeneity of a cluster C_l composed of n_l objects can be measured with the use of one of four indicators [23], see, e.g., [162]:

$$\begin{aligned}
h_1(C_l) &= \sum_{\mathbf{x}_i, \mathbf{x}_j \in C_l} d_{ij} \\
h_2(C_l) &= \max_{\mathbf{x}_i, \mathbf{x}_j \in C_l} d_{ij} \\
h_3(C_l) &= \min_{\mathbf{x}_i \in C_l} \max_{\mathbf{x}_j \in C_l} d_{ij} \\
h_4(C_l) &= \min_{\mathbf{x}_i \in C_l} \sum_{\mathbf{x}_j \in C_l, j \neq i} d_{ij}
\end{aligned} \tag{2.22}$$

The first of these indicators is the sum of dissimilarities between the pairs of objects belonging to the cluster C_l. If the elements of the set C_l are mutually sufficiently similar, the set can be treated as a clique[24], and so h_1 represents the

[23] For computational reasons, instead of distance, its squared value is often used, allowing for omitting the square root operation.

[24] In graph theory, a clique is such a subgraph, in which any two vertices are connected by an edge. Admitting that we connect with an edge the vertices that are little dissimilar, we treat a clique as a set of mutually similar vertices.

weight of a clique. The second indicator is the maximum value of dissimilarity in the group C_l; it corresponds to the diameter of the set C_l. The third indicator is being referred to as the radius of the set C_l, while the fourth one is defined as a minimum sum of dissimilarities between the objects from the set C_l and its representative. This latter indicator is called *star index* or medoid.

Separation is quantified with the use of the following indicators

$$\begin{aligned} s_1(C_l) &= \sum_{\mathbf{x}_i \in C_l} \sum_{\mathbf{x}_j \notin C_l} d_{ij} \\ s_2(C_l) &= \min_{\mathbf{x}_i \in C_l, \mathbf{x}_j \notin C_l} d_{ij} \end{aligned} \tag{2.23}$$

The first of them, $s_1(C_l)$, called the cutting cost, is the sum of distances between the objects from the set C_l and the objects from outside of this set. The second one, $s_2(C_l)$ is equal the minimum dissimilarity between the elements of the set C_l and the remaining elements of the set $\mathfrak{X}$.

Based on the measures as defined above, we can quantify the quality $J(m,k)$ of the partition of the set of m elements into k disjoint groups with the use of the indicators given below:

$$\begin{aligned} J_1(m,k) &= \frac{1}{k} \sum_{i=1}^{k} \mathfrak{w}_i \\ J_2(m,k) &= \max_{i=1,\dots,k} \mathfrak{w}_i \\ J_3(m,k) &= \min_{i=1,\dots,k} \mathfrak{w}_i \end{aligned} \tag{2.24}$$

Symbol $\mathfrak{w}_i$ denotes one of the previously defined indicators of homogeneity / separability, assigned to the i-th group. If $\mathfrak{w}$ represents homogeneity, then the indicator $J_1(n,k)$ corresponds to the average homogeneity inside groups, $J_2(n,k)$ – the maximum homogeneity, and $J_3(n,k)$ – the minimum homogeneity in the partition produced. A good partition ought to – in the case of homogeneity – be characterised by the possibly low values of these indicators, while in the case of separability – by their possibly high values.

2.4.2 The task of cluster analysis in Euclidean space

Conform to the convention adopted, each object $\mathfrak{x}_i \in \mathfrak{X}$ is described by the n-dimensional vector of features, so that the set $\mathfrak{X}$ is identified with the set of m points in the n-dimensional Euclidean space. Let

$$\overline{\boldsymbol{\mu}} = \frac{1}{m} \sum_{i=1}^{m} \mathbf{x}_i \tag{2.25}$$

be the gravity centre of the set of m objects and let

$$\boldsymbol{\mu}_j = \frac{1}{|C_j|}\sum_{x_i \in C_j} \mathbf{x}_i = \frac{1}{\sum_{i=1}^m u_{ij}} \sum_{i=1}^m u_{ij}\mathbf{x}_i = \frac{1}{n_j}\sum_{i=1}^m u_{ij}\mathbf{x}_i \tag{2.26}$$

denote the gravity centre of the j-th cluster, where $n_j = |C_j| = \sum_{i=1}^m u_{ij}$ is the cardinality of the j-th cluster.

We define two matrices (see, e.g., [121], [213]):

$$W = \sum_{i=1}^m \sum_{j=1}^k u_{ij}(\mathbf{x}_i - \boldsymbol{\mu_j})(\mathbf{x}_i - \boldsymbol{\mu_j})^{\mathsf{T}} \tag{2.27}$$

$$B = \sum_{j=1}^k \left(\sum_{i=1}^m u_{ij}\right)(\boldsymbol{\mu}_j - \overline{\boldsymbol{\mu}})(\boldsymbol{\mu}_j - \overline{\boldsymbol{\mu}})^{\mathsf{T}} \tag{2.28}$$

Matrix W is the in-group covariance matrix, while B is the inter-group covariance matrix. Both matrices together constitute a decomposition of the dispersion matrix, or variance-covariance matrix

$$T = \sum_{i=1}^m (\mathbf{x}_i - \overline{\boldsymbol{\mu}})(\mathbf{x}_i - \overline{\boldsymbol{\mu}})^{\mathsf{T}} \tag{2.29}$$

i.e.: $T = W + B$.

The typical objective functions, which are used as the criteria of selection of the proper partition, are [121]:

(a) Minimisation of the trace of matrix W. This criterion is equivalent to minimisation of the sum of squares of the Euclidean distances between the objects and the centres of clusters, to which these objects belong, that is

$$\begin{aligned} J_1(m,k) &= \sum_{j=1}^k \sum_{i=1}^m u_{ij}\|\mathbf{x}_i - \boldsymbol{\mu}_j\|^2 \\ &= \sum_{j=1}^k \frac{1}{n_j} \sum_{\mathbf{x}_i, \mathbf{x}_l \in C_j} \|\mathbf{x}_i - \mathbf{x}_l\|^2 \end{aligned} \tag{2.30}$$

In other words, minimisation of the indicator J_1 is equivalent to minimisation of the criterion of homogeneity, $h_1(C_j)/n_j$. Such approach favours spherical clusters.

(b) Minimisation of the determinant of matrix W. This criterion is useful in the situations, when the natural clusters are not spherical.

(c) Maximisation of the trace of matrix BW^{-1}. This is a generalisation of the Mahalanobis distance (2.7) for the case of more than two objects. The shortcoming of this criterion is its sensitivity to scale. Grouping obtained from the raw data may drastically differ from the one obtained after the data are rescaled (e.g. through standardisation or normalisation).

2.4.2.1 Minimising the trace of in-group covariance

Despite the limitations, mentioned before, the criterion (2.30) is among the most willingly and most frequently used in practice. We shall soon see that this is not so much the effect of its simplicity, as – surprisingly – its relatively high degree of universality.

Minimisation of the quality index (2.30) leads to the mathematical programming problem of the form

$$\begin{aligned} \min_{u_{ij}\in\{0,1\}} \sum_{i=1}^{m}\sum_{j=1}^{k} \left\| \mathbf{x}_i - \frac{\sum_{l=1}^{m} u_{lj}\mathbf{x}_l}{\sum_{l=1}^{m} u_{lj}} \right\|^2 & \\ \text{subject to} \sum_{i=1}^{m} u_{ij} > 1,\ j = 1,\ldots,k & \\ \sum_{j=1}^{k} u_{ij} = 1,\ i = 1,\ldots,m & \end{aligned} \tag{2.31}$$

It is a 0/1 (binary) programming problem with a nonlinear objective function, [162], [8]. The integer constraints, along with the nonlinear and non-convex objective function make the problem (2.31) $\mathcal{NP}$-hard. For this reason the determination of the minimum of the function J_1 is usually performed with the use of the heuristic methods. Yet, attempts have been and are being made, aiming at a satisfactory solution to the problem (2.31). A survey of these attempts can be found, for instance, in [8], [20], [282]. The goal of such studies is not only to find the optimum solution, but also to gain a deeper insight into the very nature of the task of grouping the objects. We provide below several equivalent forms of the problem (2.31), enabling its generalisation in various interesting ways.

Let $F \in \mathbb{R}^{m\times m}$ be a matrix having the elements

$$f_{ij} = \begin{cases} \frac{1}{n_j} & \text{if } (\mathbf{x}_i, \mathbf{x}_j) \in C_j \\ 0 & \text{otherwise} \end{cases} \tag{2.32}$$

where, as before, n_j denotes the number of objects in the group C_j.

If we number the objects from the set $\mathfrak{X}$ in such a manner that the first n_1 objects belong to group C_1, the successive n_2 of objects belong to group C_2, etc., then F is a block-diagonal matrix, $F = \text{diag}(F_1,\ldots,F_k)$. Each block F_j is an $n_j \times n_j$ matrix having elements $1/n_j$, i.e. $F_j = (1/n_j)\mathbf{e}\mathbf{e}^{\mathrm{T}}$, $j = 1,\ldots,k$.

Lemma 2.4.1 *Matrix F of dimensions $m \times m$, having elements defined as in equation (2.32), displays the following properties:*

(a) it is a non-negative symmetric matrix, that is, $f_{ij} = f_{ji} \geq 0$, for $i,j = 1,\ldots,m$,
(b) it is a doubly stochastic matrix, that is, $F\mathbf{e} = F^T\mathbf{e} = \mathbf{e}$,
(c) $FF = F$ (idempotency)
(d) $tr(F) = k$,

(e) *spectrum of the matrix F, $\sigma(F) = \{0, 1\}$, and there exist exactly k eigenvalues equal 1.*

Proof: *Properties (a) – (d) are obvious. We shall be demonstrating only the property (e). Since every block has the form $F_j = \mathbf{e}\mathbf{e}^T/n_j$, then exactly one eigenvalue of this submatrix is equal 1, while the remaining $n_j - 1$ eigenvalues are equal zero. The spectrum of the matrix F, $\sigma(F) = \bigcup_{j=1}^{k} \sigma(F_j)$, hence F has exactly k eigenvalues equal 1.* □

If $X = (\mathbf{x}_1 \dots \mathbf{x}_m)^{\mathsf{T}}$ is the matrix of observations, then

$$M = FX$$

is the matrix, whose i-th row represents the gravity centre of the group, to which the i-th object belongs.

Given the above notation, the quality index (2.30) can be written down in the equivalent matrix form

$$\begin{aligned} J_1(C_1, \dots, C_k) &= \sum_{j=1}^{k} \sum_{x_i \in C_j} \|\mathbf{x}_i - \boldsymbol{\mu}_j\|^2 \\ &= \operatorname{tr}\big((X - M)^{\mathsf{T}}(X - M)\big) \\ &= \operatorname{tr}(X^{\mathsf{T}}X + M^{\mathsf{T}}M - 2X^{\mathsf{T}}M) \end{aligned}$$

Taking advantage of the additivity and commutativity of the matrix trace, see properties (b) and (c) in page 232, as well as symmetry and idempotence of matrix F, we transform the above expression to the form

$$\begin{aligned} J_1(C_1, \dots, C_k) &= \operatorname{tr}(X^{\mathsf{T}}X + X^{\mathsf{T}}F^{\mathsf{T}}FX - 2X^{\mathsf{T}}FX) \\ &= \operatorname{tr}(X^{\mathsf{T}}X + X^{\mathsf{T}}FX - 2X^{\mathsf{T}}FX) \\ &= \operatorname{tr}(X^{\mathsf{T}}X - X^{\mathsf{T}}FX) \\ &= \operatorname{tr}(X^{\mathsf{T}}X - X^{\mathsf{T}}XF) \end{aligned}$$

Let

$$K = X^{\mathsf{T}}X$$

It is a symmetric and non-negative matrix. The expression $\operatorname{tr}(X^{\mathsf{T}}XF) = \operatorname{tr}(KF) = \sum_{ij} k_{ij} f_{ij}$ is a linear combination of the elements of matrix K, hence, in case of the nonlinear objective function J_1 we obtain a linear function! Finally, the task of grouping of objects in the Euclidean space reduces to either *minimisation* of the indicator

$$J_1(C_1, \dots, C_k) = \operatorname{tr}\big(K(\mathbb{I} - F)\big) \tag{2.33}$$

or, equivalently, if we ignore the constant component K, to *maximisation* of the indicator

$$J_1'(C_1, \dots, C_k) = \operatorname{tr}(KF) \tag{2.34}$$

The first of these forms is used, in particular, by Peng and Wei in [282], while the second, by, for instance, Zass and Shashua in [381]. We concentrate on the task of maximisation[25]

$$\begin{array}{c} \max_{F \in \mathbb{R}^{m \times m}} \mathrm{tr}\,(KF) \\ \text{subject to } F \geq 0, F^{\mathrm{T}} = F, F\mathbf{e} = \mathbf{e} \\ F^2 = F, \mathrm{tr}\,(F) = k \end{array} \tag{2.35}$$

The above formulation allows for the generalisation of the task of grouping in a variety of manners:

(a) Minimisation of the indicator (2.30) assumed that the observations are points in n-dimensional Euclidean space. In such a case the prototypes are sought, being the gravity centres of the possibly compact groups. The quality index of the form of $\mathrm{tr}\,(KF)$ makes it possible to replace the distances between the points by a more general kernel function[26]. This shifts our interest in the direction of the so-called relational grouping, where, instead of the Euclidean distance between the pairs of points, a more general measure of similarity is being used, $k_{ij} = s(\mathbf{x}_i, \mathbf{x}_j)$, e.g. the Gaussian kernel function $k_{ij} = \exp\big(-\gamma\|\mathbf{x}_i - \mathbf{x}_j\|^2\big)$, where $\gamma > 0$ is a parameter. Thereby we give up the assumption that the set $\mathfrak{X}$ has to have the representation $X \in \mathbb{R}^{m \times n}$.

(b) The objective function, appearing in the problem (2.35) can be replaced by the Bregman divergence (see definition 2.2.1 on page 31) D_ϕ, this leading to the problem, see [358]

$$\begin{array}{c} \max_{F \in \mathbb{R}^{m \times m}} D_\phi(K, F) \\ \text{subject to } F \geq 0, F^{\mathrm{T}} = F, F\mathbf{e} = \mathbf{e} \\ F^2 = F, \mathrm{tr}\,(F) = k \end{array} \tag{2.36}$$

When $\phi = r^2$, problem (2.36) is reduced to problem (2.35). The generalised variant of the k-means algorithm for such a case is considered in the study [37]. It is shown there that, in particular, the prototypes of groups are determined as the (weighted) gravity centres of these groups, and, even more importantly, that such a definition of the prototype is correct only if the dissimilarity of the objects is measured through some Bregman divergence.

Let us also note here that formulation (2.35) turns our attention towards the doubly stochastic matrices, that is – such symmetric and non-negative matrices

[25] Peng and Wei note in [282] that the idea of representing the objective function appearing in the problem (2.31) in the form of minimisation of the trace of an appropriate matrix was forwarded first by A.D. Gordon and J.T. Henderson in their paper, entitled ,,An algorithm for Euclidean sum of squares", which appeared in the 33rd volume of the journal *Biometrica* (year 1977, pp. 355-362). Gordon and Henderson, though, wrote that this idea had been suggested to them by the anonymous referee!

[26] See Section 2.5.4.

that the sum of every row and every column is equal 1. One can find interesting remarks on applications of such matrices in, for instance, [208].

In order to enhance the flexibility of the formulation (2.35), let us replace the matrix F by the product GG^{T}, where G is the matrix of the dimensions $m \times k$, having the elements

$$g_{ij} = \begin{cases} \frac{1}{\sqrt{n_j}} & \text{if } (\mathbf{x}_i, \mathbf{x}_j) \in C_j \\ 0 & \text{otherwise} \end{cases}$$

Matrix G fulfills the following conditions: (a) it is non-negative, $g_{ij} \geq 0$, $i = 1, \dots, m$, $j = 1, \dots, k$, (b) $G^{\mathrm{T}}G = \mathbb{I}$, (c) $GG^{\mathrm{T}}\mathbf{e} = \mathbf{e}$. The non-negativity of the matrix G means that F is a completely positive matrix, and its cp-rank[27] equals k.

By referring to the fact that $\operatorname{tr}(AB) = \operatorname{tr}(BA)$, provided both products do exist, we can turn the task of maximisation (2.35) into the one of the form

$$\begin{aligned} &\max_{G \in \mathbb{R}^{m \times k}} \operatorname{tr}(G^{\mathrm{T}}KG) \\ \text{subject to } &G \geq 0, \\ &G^{\mathrm{T}}G = \mathbb{I}, \\ &GG^{\mathrm{T}}\mathbf{e} = \mathbf{e} \end{aligned} \tag{2.37}$$

Taking into account the formulation (2.37) we can treat the problem of grouping as a search for such a matrix F, for which $\operatorname{tr}(KF)$ attains the maximum in the set of all matrices $\mathbb{R}^{m \times m}$ fulfilling two additional conditions: (a) F is a doubly stochastic matrix, (b) F is a completely positive matrix, and its cp-rank $= k$, that is: $F = GG^{\mathrm{T}}$, where G is a non-negative matrix of the dimensions $m \times k$. Zass and Shashua propose in [381] a two-stage procedure:

(i) The given matrix K is replaced by the doubly stochastic matrix $\widetilde{F}$. In order to do this, one can use the Sinkhorn-Knopp method, which consists in the intermittent normalisation of rows and columns of matrix K. Other, more effective methods of turning a matrix into the doubly stochastic form are commented upon by Knight in [208].
(ii) In the set of the non-negative matrices of dimensions $m \times k$ such a matrix G is sought, which minimises the error $\|\widetilde{F} - GG^{\mathrm{T}}\|_F$, where $\| A\|_F = \sqrt{\operatorname{tr} A^{\mathrm{T}}A} = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{m} a_{ij}^2}$ denotes the Frobenius norm.

The concepts here barely outlined have been developed into the advanced methods of cluster analysis. A part of them makes use of the so-called semi-definite programming; we can mention here the studies, reported in [282], [71], [218], or in [371]. They concern not only the typical problems of minimising the trace of an appropriate matrix, but also more advanced methods of grouping, which are considered in the further parts of this book.

[27] See definition B.2.4 in page 232.

2.4.2.2 Approximating the data matrix

The quality index (2.30) can be transformed to

$$J_1 = \|X - UM\|_F^2 \tag{2.38}$$

where $M \in \mathbb{R}^{k \times n}$ is the matrix with group centroids being its rows, $M = (\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_k)^{\mathsf{T}}$, while $U \in \mathbb{R}^{m \times k}$ is the matrix indicating the assignment of the i-th object to the j-th group, $U = (\mathbf{u}_1, \dots, \mathbf{u}_m)^{\mathsf{T}}$.

Minimisation of the indicator (2.38) allows for taking a different perspective on the task of grouping: we look for a possibly good approximation of the data matrix by the product of two matrices, U and M. If $u_{ij} \in \{0, 1\}$, then this task is being carried out with the help of the following procedure:

(a) If $\widehat{M} = (\widehat{\boldsymbol{\mu}}_1, \dots, \widehat{\boldsymbol{\mu}}_k)^{\mathsf{T}}$ is the current approximation of the matrix M, then the elements $\widehat{u}_{ij}$ of the matrix $\widehat{U}$, constituting the approximation of U, have the form

$$\widehat{u}_{ij} = \begin{cases} 1 \text{ if } j = \underset{1 \le t \le k}{\arg\min}\ \|\mathbf{x}_i - \widehat{\boldsymbol{\mu}}_t\|_F^2 \\ 0 \text{ otherwise} \end{cases} \tag{2.39}$$

(b) If $\widehat{U}$ is the current approximation of the matrix U, then determination of the matrix $\widehat{M}$ minimising the indicator J_1 is a classical problem of regression. From the condition[28] $\partial J_1 / \partial \widehat{M} = \widehat{U}^{\mathsf{T}}(\widehat{U}\widehat{M} - X) = 0$ we get

$$\widehat{M} = (\widehat{U}^{\mathsf{T}}\widehat{U})^{-1}\widehat{U}^{\mathsf{T}} X \tag{2.40}$$

(c) Steps (a) and (b) are repeated until the terminal condition is fulfilled, this condition consisting in the performance of a given number of repetitions, or in stabilisation of the elements of the matrix $\widehat{U}$.

The algorithm is initiated by specifying either an approximation $\widehat{U}$, or the matrix $\widehat{M}$. A better variant, ensuring faster convergence, is to start from the matrix $\widehat{M}$. The methods of its initialisation are considered in Section 3.1.3.

Minimisation of the indicator (2.38) constitutes, in its essence, the problem of the so-called non-negative factorisation, playing an important role in machine learning, [359], bioinformatics, [95], text analysis, [48], [374], or in recommender systems [211]. Yet, the problem (2.38) differs somewhat from the classical formulation [228], where it is required to have both matrices non-negative. In the case of grouping, matrix M does not have to be non-negative, while matrix U must satisfy certain additional constraints, like, e.g., $\sum_{j=1}^{k} u_{ij} = 1$. The thus formulated task of factorisation of matrix X is a subject of intensive studies, e.g. [103], [100], [231], [230], or [184].

By generalising the indicator (2.38) to the form

[28] We take advantage here of the fact that $\|A\|_F^2 = \operatorname{tr}(A'A)$.

$$J_1 = \|X - U^{\alpha} M^{\mathrm{T}}\|_F^2 \tag{2.41}$$

where $\alpha > 1$, we obtain the formulation leading to fuzzy grouping that we consider in Section 3.3. An example of application of this technique in bioinformatics shall be presented in Section 2.5.5.

2.4.2.3 Iterative algorithm of finding clusters

The algorithms of determination of the partition of objects into k classes are usually iterative procedures, which converge at a local optimum, [160]. An instance thereof has been presented in the preceding section. Its weak point is the necessity of performing operations on matrices in step (b). Note, though, that due to a special structure of the matrix of assignments U:

(a) Product $\widehat{U}^{\mathrm{T}}\widehat{U} = \widetilde{U}$ is a diagonal matrix having elements

$$\widetilde{u}_{jj} = \sum_{i=1}^{m} \widehat{u}_{ij} = n_j, \quad j = 1, \ldots, k$$

where n_j denotes the number of elements of the j-th group. Hence, $\widetilde{U}^{-1}$ is a diagonal matrix, as well, having elements $\widetilde{u}_{jj}^{-1} = 1/n_j$.

(b) Matrix $\widetilde{M} = \widehat{U}^{\mathrm{T}} X$ has the dimensions $k \times n$, and its i-th row is the sum of rows of the matrix X, corresponding to the elements of the i-th cluster. So, the i-th row of the matrix $\widetilde{U}^{-1}\widetilde{M}$ is the arithmetic mean of the coordinates of objects, assigned to the i-th group.

The general form of the iterative procedure of assigning objects to clusters is shown in the pseudocode 2.2. Here, the weights are additionally used, indicating the contribution of a given object to the relocation of the gravity centres. This mechanism was introduced by Zhang [383], and was applied, in particular, by Hamerly and Elkan, [160]. When all weights are equal 1, the algorithm 2.2 corresponds to the algorithm from the preceding section and represents the classical Lloyd's heuristics [238].

The essence of the algorithm is the iterative modification of the assignment of objects to clusters. It is most common to assign an object to the cluster with the closest gravity centre, i.e.

$$u_{ij} = \begin{cases} 1 \text{ if } j = \arg\min\limits_{1 \le t \le k} \|\mathbf{x}_j - \boldsymbol{\mu}_t\| \\ 0 \text{ otherwise} \end{cases} \tag{2.43}$$

where $\boldsymbol{\mu}_t$ denotes the gravity centre of cluster C_t (this rule was applied in step (a) of the algorithm from the preceding section). It is the rule *the winner takes all* – known also from the theory of competitive supervised learning[29]. The clusters,

[29] See, e.g., J. Hertz, A. Krogh, R.G. Palmer: *Introduction to the Theory of Neural Computation. Santa Fe Institute Series*, Addison-Wesley, 1991.

Algorithm 2.2 Iterative algorithm of cluster analysis (generalised Lloyd's heuristics)

```
Require: Data set $X$ and number of groups $k$.
Ensure: Gravity centres of classes $\{\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_k\}$ along with the assignment of objects
    to classes $U = [u_{ij}]_{m \times k}$.
1: Initialisation. Select the gravity centres of clusters and assign weights to objects
    $w(\mathbf{x}_i)$.
2: Updating of assignments: for each object determine its assignment to a cluster and,
    possibly, also its weight.
3: Updating of the gravity centres of clusters:
    $$\boldsymbol{\mu}_j = \frac{\sum_{i=1}^{m} u_{ij} w(\mathbf{x}_i)\mathbf{x}_i}{\sum_{i=1}^{m} u_{ij} w(\mathbf{x}_i)} \qquad (2.42)$$
4: Repeat steps 2 and 3 until the stopping condition is fulfilled, usually assumed to
    be the lack of changes in the assignment of objects to clusters.
```

determined in this manner, are called Voronoi clusters[30]. Formally, if $\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_k$ is a set of prototypes, then the Voronoi cluster W_j is the set of points, for which $\boldsymbol{\mu}_j$ is the closest prototype, that is:

$$W_j = \{\mathbf{x} \in \mathbb{R}^n | j = \arg\min_{1 \le l \le k} \|\mathbf{x} - \boldsymbol{\mu}_l\|\} \tag{2.44}$$

These clusters are convex sets, i.e.

$$[(\mathbf{x}' \in W_j) \wedge (\mathbf{x}^* \in W_j)] \Rightarrow [\mathbf{x}' + \alpha(\mathbf{x}^* - \mathbf{x}')] \in W_j, \quad 0 \le \alpha \le 1$$

The division of the space $\mathbb{R}^n$ into Voronoi sets is called Voronoi tessellation (tiling) or Dirichlet tessellation. Their nature is illustrated in Figure 2.6. In the two-dimensional case the lines, separating the regions, belonging to different clusters, are the lines of symmetry of the segments, linking the neighbouring gravity centres. Although effective algorithms for tiling are known for the two-dimensional case [286], the very notion remains useful also in the n-dimensional case.

2.4.3 Grouping according to cluster volume

Minimisation of the trace or the determinant of the matrix W leads to clusters having similar numbers of elements. Besides, minimisation of $\operatorname{tr}(W)$ favours spherical clusters.

[30] Georgiy Fedosiyevich Voronoi, whose name appears in the Voronoi diagrams, was a Russian mathematician of Ukrainian extraction. He lived in the years 1868 – 1908. Interesting information on this subject can be found in 17th chapter of popular book by Ian Stewart, entitled *Cows in the Maze. And other mathematical explorations*, published in 2010 by OUP Oxford.

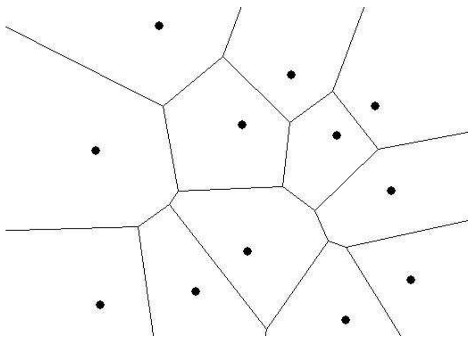


Fig. 2.6. Voronoi tessellation, that is, the boundaries of clusters determined by the gravity centres marked by dark dots.

Report [313] presents the MVE algorithm (*Minimum Volume Ellipsoids*), in which clusters are represented by the hyper-ellipsoids having minimum volume. In distinction from the algorithms outlined in the preceding section, the MVE algorithm is independent of scale and allows for the determination of clusters having different numbers of elements. The dissimilarity measure, applied in this algorithm, is related to Mahalanobis distance. A similar issue is also discussed by Kumar and Orlin in [220].

The quality criterion adopted has the following form:

$$\begin{array}{rl} & \min \sum_{j=1}^{k} \mathrm{vol}\,(C_j) \\ \text{subject to} & \sum_{j=1}^{k} |C_j| \geq (1-\alpha)m,\ \ 0 \leq \alpha < 1 \\ & C_j \subset X \end{array}$$

The first limiting condition means that existence of at most αm outliers in the data set is allowed, while the remaining observations have to be assigned to k clusters.

The hyper-ellipsoid E_j, containing the objects from the group C_j is defined by its centre $\mathbf{c}_j$ and the symmetric and positive definite matrix Q_j, e.g. the covariance matrix, characterising this group of data. So,

$$E_j = \{\mathbf{x} \in \mathbb{R}^n \colon (\mathbf{x} - \mathbf{c}_j)^{\mathrm{T}} Q_j^{-1} (\mathbf{x} - \mathbf{c}_j) \leq 1\} \tag{2.45}$$

Its volume is equal $\sqrt{\det(Q_j)}$. Taking advantage of this fact, we can reduce the task of partitioning the set X into k groups to the problem of semi-definite programming of the form (see, e.g., [282])

$$\begin{aligned}
&\min \sum_{j=1}^{k} \sqrt{\det(Q_j)} \\
\text{subject the } &(\mathbf{x}_i - \mathbf{c}_j)^{\mathtt{T}} Q_j^{-1} (\mathbf{x}_i - \mathbf{c}_j), \forall(\mathbf{x}_i \in C_j), j = 1, \dots, k \\
&\textstyle\sum_{j=1}^{k} |C_j| \geq (1-\alpha)m,\ 0 \leq \alpha < 1 \\
&C_j \subset X \\
&C_j \succ 0,\ j = 1, \dots, k
\end{aligned} \tag{2.46}$$

Symbol $C_j \succ 0$ means that C_j is a symmetric and positive definite matrix.

Publication [220] presents two algorithms that solve the problem formulated above: (1) `kVolume`, an iterative algorithm, which divides up the set of observations into the predefined number of groups, and (2) `hVolume`, that is – a hierarchical grouping algorithm, which generates a monotonic family of clusters. In both cases a fast algorithm of calculating volumes is made use of, as presented in [330].

2.4.4 Generalisations of the task of grouping

In many instances, such as: grouping of documents, social network analysis, or bioinformatics, particular objects may belong simultaneously to several groups. In other words, instead of partitioning the data set, we look for the covering of this set. One of the ways to deal with such situations is to apply the algorithms of fuzzy clustering, for instance the FCM algorithm from Section 3.3. Yet, in recent years, emphasis is being placed on the algorithms allowing not only for an explicit reference to simultaneous assignment of objects to various groups, but also making it possible to identify the optimum coverings of the given set of objects. Their detailed consideration exceeds the assumed framework of this book. We shall only mention a couple of interesting solutions, encouraging the Reader to an own study of this matter:

– Banerjee *et al.* presented in [36] `MOC` – *Model based Overlapping Clustering*, which can be seen as the first algorithm, which produces the optimum covering of the data set. The authors mentioned make use of the probabilistic relational model[31], proposed for purposes of analysis of the microarrays. While the original solution concentrates on the normal distributions, `MOC` operates on arbitrary distributions from the exponential family. Besides, by introducing Bregman divergence, one can apply here any of the distances, discussed in Section 2.2.1. In the opinion of the respective authors, this new algorithm can be applied in document analysis, recommender systems, and in all situations, in which we deal with highly dimensional and sparse data.

[31] E. Segal, A. Battle, and D. Koller. Decomposing gene expression into cellular processes. In: *Proc. of the 8th Pacific Symp. on Biocomputing* (PSB), 2003, pp. 89-100

- Cleuziou[32] proposed `OKM` – *Overlapping k-Means*, which is a generalisation of the k-means algorithm. This idea was then broadened to encompass the generalised k-medoids algorithm.
- In other approaches to the search for the coverings, graph theory and neural networks are being applied. In the case of the graph theory methods, first the similarity graph is constructed (analogously as in the spectral data analysis, considered in Section 5), and then all the cliques contained in it are sought[33].

A survey of other algorithms is provided also in [283].

Grouping of objects in highly dimensional spaces on the basis of distances between these objects gives rise to problems discussed in Section 2.2.1.1. A classical solution consists in projecting the entire data set on a low dimensional space (by applying, for instance, multidimensional scaling, random projections, or principal component analysis) and using some selected algorithm to cluster the thus obtained transformed data. In practice, though, it often turns out that various subsets of data (”clusters”) may be located in different subspaces.

That is why the task of grouping is defined somewhat differently. Namely, such a breakdown $C_1, \dots, C_k$ of the data set is sought, along with the corresponding subsets of features, $F_i \subset \{1, \dots, n\}$, that points assigned to C_j be sufficiently close one to another in the F_j dimensional space. It can be said that C_j is composed of points which, after projection on the F_j dimensional space, constitute a separate cluster. A survey of methods meant to solve the thus formulated task is provided in [279], [214].

2.5 Other methods of cluster analysis

Methods, which have been outlined in the two preceding sections belong to two essential streams developing within cluster analysis. In both cases a cluster is understood as a set of objects that are mutually much more similar than any two objects selected from two different clusters.

2.5.1 Relational methods

It has been assumed till now that each object is represented by a vector of feature values. An alternative description is constituted by the relation of similarity or dissimilarity for the pairs of objects. We encounter such situation in social sciences or in management, [91], [168]. By operating on the relation of similarity / dissimilarity we can conceal the values of the attributes, which may be of significance in such domains as, for instance, banking. It also is simpler to deal

[32] G. Cleuziou. A generalization of k-means for overlapping clustering. Université d'Orléans , LIFO, Rapport No RR-2007-15.

[33] See, e.g., W. Didimo, F. Giordano, G. Liotta. Overlapping cluster planarity. In: Proc. APVIS 2007,2007, p. 73-80, M. Fellows, J. Guo, C. Komusiewicz, R. Niedermeier, J. Uhlmann. Graph-based data clustering with overlaps, *COCOON*, 2009, p.516-526

with mixed attribute types (both quantitative and qualitative). The sole problem to be solved at this level is the choice of the function measuring similarity / dissimilarity of objects.

The notion of ”relational methods” is sometimes used in a wider context. Thus, for instance, a set of relations may be given, S_i, defined on different subsets $\mathfrak{X}_i$ of objects, [18], or relations may have a more complex character. In particular, when grouping documents, it is worthwhile to consider relations between subject groups and keywords.

2.5.2 Graph and spectral methods

The set X is often identified with the set of vertices of a certain graph G, whose edges represent the connections between the objects; e.g. the pair $\{\mathbf{x}_i, \mathbf{x}_j\}$ is an edge in G, if the two objects are similar in the degree not lower than s_τ. In such a context a cluster becomes equivalent to, for instance, a clique, that is, a connected subgraph of the graph G, i.e. such a one that every two vertices of the subgraph are connected by an edge. In another definition it is assumed that a cluster is such a subgraph G_i, whose vertices communicate exclusively among themselves, and do not communicate with other vertices, outside this subgraph. This means that when extracting clusters we take into account their connectivity and not only their compactness. This is illustrated on Fig. 2.7. Using pairwise distances between the points from the left panel we construct a graph shown on the right panel. Here, each node is linking with other five nearest nodes.

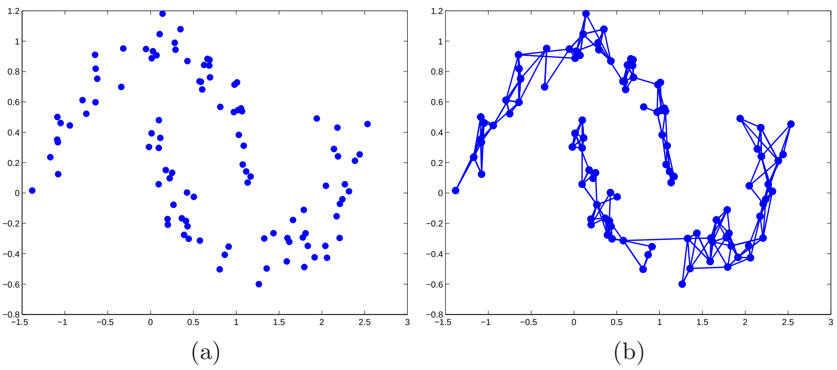


Fig. 2.7. Identification of clusters by examining mutual similarity between objects: (a) a set of points[35], (b) a graph obtained by joining five nearest neighbors of each node

The progenitor of graph cut clustering is Zahn’s[36] approach, consisting of two steps. First, using similarity matrix describing relationships among the objects, a maximum spanning tree is constructed, and then the edges with small weights

[36] C.T. Zahn, Graph-theoretic methods for detecting and describing gestalt clusters. *IEEE Trans. Comput.*, 20(1):68-86, 1971.

are removed from this tree to get a set of connected components. This method is successful in detecting clearly separated clusters, but if the density of nodes is changed, its performance will deteriorate. Another disadvantage is that the cluster structure must be known in advance.

The understanding of clusters, as described above, places the problem of their extraction in the context of graph cutting. In particular, when we assign to every edge $\{v_i, v_j\}$ in the graph G a similarity degree s_{ij}, the problem boils down to removing the edges with small weights in order to decompose G into connected components. Such a procedure ensures that the sum of weights of the edges, connecting vertices from different groups is small in comparison with the sum of weights linking the vertices, belonging to the same subgraph [37]. An exhaustive survey of techniques, which are applied in the partitioning of graphs, can be found e.g. in [127] or [305].

A particularly important domain of application of this group of algorithms is parallel computing. Assume that solving a certain problem requires performing of m tasks, each of which constitutes a separate process, program or thread, realised by one of c processors. When the processors are identical, and all tasks are of similar complexity, then we can assign to every processor the same number of tasks. Usually, though, realisation of a concrete task requires the knowledge of partial results, produced by other tasks, which implies the necessity of communication between these tasks. In this manner we obtain a graph, with vertices corresponding to individual tasks, while edges – correspond to their communication needs. Communication between the processors, which form a parallel computing machine is much slower than data movement within one single processor. In order to minimise communication to a necessary level, the processes (vertices of the graph) ought to be partitioned into groups (i.e. assigned to processors) in such a way that the number of edges, linking different groups, be minimal. This is the formulation of a problem, whose solutions are considered, in particular, in [127].

Another group of problems is constituted by image segmentation. Here, an image is modelled as an undirected weighted graph, its vertices being pixels, or groups of pixels, while the weights of edges correspond to the degree of similarity (or dissimilarity) between the neighbouring pixels. This graph (image) is to be segmented conform to a certain criterion, defining "good" clusters.

An interesting offer, allowing for identification of complex cluster structures is spectral graph theory – its application in clustering is considered in detail in this book in Chapter 5. Various methods of spectral cluster analysis are reviewed in [124], [351] and [305]. It is worth noting that spectral methods play nowadays a significant role in practical tasks of data analysis and mining, such as information retrieval [49], bioinformatics [174], or recommender systems [1], [221].

[37] which means that objects, belonging to the same class are mutually sufficiently similar, while objects, belonging to different groups are sufficiently mutually dissimilar.

2.5.3 Density-based methods

In a different perspective, a cluster is conceived as a dense area in the space of objects, surrounded by low density areas. This kind of definition is convenient in situations, when clusters have irregular shapes, and the data set contains outliers and noisy observations – see Figure 2.8. A typical representative of this group of algorithms is DBSCAN [120] and its later modifications [299], [119] and [17].

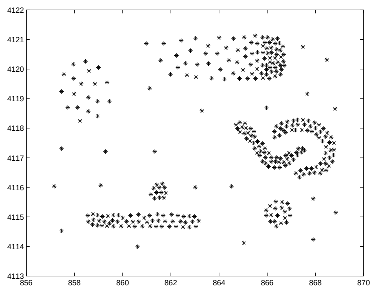


Fig. 2.8. An example of clusters with irregular shapes and various densities. The data set contains also *outliers*.

Before presenting the essential idea of this algorithm, we shall introduce the necessary definitions.

(a) Let $d(\mathbf{x}, \mathbf{y})$ denote distance between any two points $\mathbf{x}, \mathbf{y} \in X$ and let $\epsilon > 0$ be a parameter. The ϵ-neighbourhood of the object $\mathbf{x}$ is the set

$$N_\epsilon(\mathbf{x}) = \{\mathbf{x}' \in X : d(\mathbf{x}, \mathbf{x}') \leq \epsilon\}$$

(b) An object $\mathbf{x} \in X$ is called the internal point of a cluster, if its ϵ-neighbourhood contains at least *minPts* objects, i.e. when $|N_\epsilon(\mathbf{x})| \geq minPts$, where *minPts* is a parameter.
(c) An object $\mathbf{x} \in X$ is called a border point, if $|N_\epsilon(\mathbf{x})| < minPts$, but this neighbourhood contains at least one internal point.
(d) If $\mathbf{x} \in X$ is neither an internal point, nor a border point, then it is treated as a disturbance (*outlier*).

In construction of the particular clusters use is made of the notion of density-reachability . Namely, a point $\mathbf{y} \in X$ is *directly* density-reachable from the point $\mathbf{x} \in X$ if $\mathbf{y} \in N_\epsilon(\mathbf{x})$, and, besides, $\mathbf{x}$ is an internal point, that is, it is surrounded by a sufficiently high number of other points. Note that the relationship of direct density-reachability is assymetric. Then, $\mathbf{y}$ is called density-reachable from the point $\mathbf{x}$ if there exists a sequence $\mathbf{x}_1, \ldots, \mathbf{x}_n$ of points such that $\mathbf{x}_1 = \mathbf{x}$, $\mathbf{x}_n = \mathbf{y}$, and each point $\mathbf{x}_{i+1}$ is directly density-reachable from $\mathbf{x}_i$, $i = 1, \ldots, n-1$. Note that the thus defined relation of reachability is asymmetric ($\mathbf{y}$ may be a border point). That is why the notion of density-connectedness is introduced : points

$\mathbf{x}, \mathbf{y} \in X$ are density-connected, if there exists such a point $\mathbf{z} \in X$ that both $\mathbf{x}$ and $\mathbf{y}$ are density-reachable from $\mathbf{z}$. Clusters, generated by the DBSCAN algorithm have the following properties:

(i) All points, belonging to a cluster are mutually density-connected.
(ii) If an internal point is density-connected with another point of a cluster, then it is also an element of this cluster.

Generation of clusters, having such properties, is outlined here through the pseudocode 2.3. Note that we make use here only of the internal and border points. Each two internal points, whose mutual distance does not exceed the value of ϵ are put in the same cluster. On the other hand, the border points are classified in any cluster, provided a neighbour of this point is an internal point of the cluster.

Algorithm 2.3 DBSCAN algorithm

```
1: Initialisation. Mark the points from the set $X$ as internal, border or noisy points.
2: Remove the noisy points.
3: Connect with an edge the neighbouring internal points (i.e. the internal points
   situated at a distance not bigger than $\epsilon$).
4: Form a cluster out of the neighbouring internal points.
5: Assign the border points to one of the clusters, upon which they neighbour.
```

Choice of the appropriate value of the radius ϵ influences significantly the results of the algorithm. If ϵ is too big – density of each point is identical and equal m (that is – the cardinality of the set X). If, however, the value of the radius is too small, then $|N_\epsilon(\mathbf{x}) = 1|$ for any $\mathbf{x} \in X$. The value of the parameter ϵ is often assumed as the so-called k-distance, $k\text{-}dist(\mathbf{x})$, namely the distance between the point $\mathbf{x}$ and its k-th nearest neighbour. In the case of the two-dimensional sets the value of $k = 4$ is often assumed, although there is also a suggestion of taking $k = n + 1$.

In order to select the proper value of ϵ a diagram is developed of the increasingly ordered values of $k\text{-}dist(\mathbf{x})$ for all $\mathbf{x} \in X$. It is characterised by the appearance of a value, following which an abrupt increase of distance takes place. This is the value, which is chosen as ϵ. The procedure is illustrated in Figure 2.9. Its left part shows the data set [38], while the right part shows the diagram of $k\text{-}dist(\mathbf{x})$ for $k = 4$. It is usually assumed that $minPts = k$. One can also read out of the diagram that $\epsilon \approx 1.0$.

In a general case, instead of k-distance, one can use a certain function $g(\mathbf{x})$, characterising density at point $\mathbf{x}$. In the case of the DENCLUE algorithm [175] this is the sum of the components of the function

[38] It arose from adding 14 randomly generated points to the set, described in Chapter 6, namely `data3_2`.

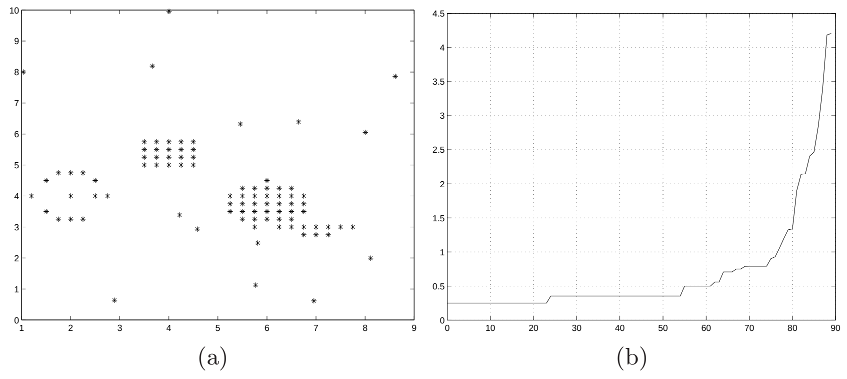


Fig. 2.9. Diagram of k-distances between the points of an exemplary data set $\mathfrak{X}$: (a) An exemplary data set composed of 90 points, (b) increasingly ordered values of the average distance between each of the 90 points of the data set $\mathfrak{X}$ and their $k = 4$ nearest neighbours.

$$g(\mathbf{x}) = \sum_{\mathbf{x}' \in X} f(\mathbf{x}, \mathbf{x}')$$

where $f(\mathbf{x}, \mathbf{x}')$ is, in principle, an arbitrary function, which describes the influence exerted by the object $\mathbf{x}'$ on the object $\mathbf{x}$, e.g.

$$f(\mathbf{x}, \mathbf{x}') = \exp\Big(- \frac{d(\mathbf{x}, \mathbf{x}')}{2\sigma^2} \Big)$$

with $\sigma > 0$ being a parameter. In this context, clusters are described as local maxima of the function $g(\mathbf{x})$.

The fundamental qualities of the DBSCAN algorithm are as follows:

1. Knowledge of the number of clusters existing in the set X is not required.
2. Clusters may have arbitrary shapes. Owing to the parameter *minPts* the effect of a single connection is reduced, as manifested by the appearance of thin lines, composed of points, belonging to different clusters.
3. Data are allowed to contain noise and outliers.
4. The algorithm requires specification of just two parameters: the radius ϵ, and the number *minPts*, used in classification of points.
5. The algorithm is only slightly sensitive to the order, in which the individual objects from the set X are considered.

Yet, there is a significant requirement that clusters have similar densities. This shortcoming is done with in the variants of the algorithm – GDBSCAN [299] and LDBSCAN [111]. Another essential weak point is the fact that the quality of the algorithm strongly depends upon the definition of distance $d(\mathbf{x}, \mathbf{y})$. We already know that, for instance, Euclidean distance loses on its usefulness in the case of analysis of highly dimensional data.

Moore [262] proposed the so-called anchor algorithm to analyse highly dimensional data – see the pseudocode 2.4. This algorithm belongs to the class of algorithms grouping on the basis of the minimum class diameter.

Algorithm 2.4 Anchor grouping algorithm, [262]

```
Require: Data set $X$, number of clusters $K$, minimum number of objects in a cluster $n_{min}$.
1: Select randomly a point. Initiate $k = 1$, $k' = 0$. Take as the anchor $a^1$ the point from the set $X$ which is most distant from the initial point.
2: Assign to the group $C_k$, initiated by the anchor $a^k$, these objects, which are closer to $a^k$ than to other anchors. The objects form a list, ordered decreasingly with respect to their distance from the anchor.
3: Check, whether the anchor $a^k$ has the sufficient number of objects assigned to it. If $|C_k| < n_{min}$, then $k' = k' + 1$.
4: Substitute $k = k + 1$. For $a^k$ substitute the point that is most distant from the remaining anchors.
5: If $k - k' < K$, then go to step 2.
6: return partition of the set $X$ into disjoint groups.
```

It should be noted that the algorithm may produce more than K groups. It may also happen that the algorithm shall not produce sets containing more than n_{min} elements.

2.5.4 Potential (kernel) function methods

These methods originate from the work of Ajzerman, Braverman and Rozonoer [6], where these authors used the term of potential function. A short introduction to this approach can be found in Section 5.6 of the monograph [337]. Along with the development of the theory of support vector machines (SVM) [347] the term "kernel function" replaced the earlier term of the "potential function". We present below, following [124], the fundamental assumptions of cluster analysis methods using the notion of kernel function.

We assume (for simplicity and in view of practical applications), that we deal with n-dimensional vectors having real-valued components (and not the complex numbers, as this is assumed in the general theory). Hence, as until now, $X = \{\mathbf{x}_1, \dots, \mathbf{x}_m\}$ denotes the non-empty set of objects, with $\mathbf{x}_i \in \mathbb{R}^n$.

Definition 2.5.1 *A function* $\mathsf{K}\colon X \times X \to \mathbb{R}$ *is called the positive definite kernel function (Mercer kernel or simply kernel) if: (i)* $\mathsf{K}(\mathbf{x}_i, \mathbf{x}_j)$ *is a symmetric function, and (ii) for any vectors* $\mathbf{x}_i, \mathbf{x}_j \in X$ *and any real-valued constants* $c_1, \dots, c_m$ *the following inequality holds:*

$$\sum_{i=1}^{m} \sum_{j=1}^{m} c_i c_j \mathsf{K}(\mathbf{x}_i, \mathbf{x}_j) \geq 0$$

□

If $\mathsf{K}_1(\mathbf{x},\mathbf{y})$, $\mathsf{K}_2(\mathbf{x},\mathbf{y})$ are kernel functions, then their sum, product, and $a\mathsf{K}(\mathbf{x},\mathbf{y})$, where $a > 0$, are also kernel functions. The typical kernel functions, which are used in machine learning are:

(a) Linear kernel $\mathsf{K}_l(\mathbf{x},\mathbf{y}) = \mathbf{x}^{\mathrm{T}}\mathbf{y} + c$. In the majority of cases, the algorithms, which use linear kernel functions, are close to equivalence with their "non-kernel" counterparts (thus, e.g., the kernel-based variant of the principal component analysis with the linear kernel is equivalent to the classical PCA algorithm).

(b) Polynomial kernel $\mathsf{K}(\mathbf{x},\mathbf{y}) = (\alpha\mathbf{x}^{\mathrm{T}}\mathbf{y} + c)^d$, where α, c and the degree of the polynomial, d, are parameters. The polynomial kernel functions are applied, first of all, in the situations, in which normalised data are used.

(c) Gaussian kernel $\mathsf{K}(\mathbf{x},\mathbf{y}) = \exp\Big(- \|\mathbf{x}-\mathbf{y}\|^2/(2\sigma^2)\Big)$, where $\sigma > 0$ is a parameter, whose choice requires special care. If its value is overestimated, the exponent will behave almost linearly, and the very nonlinear projection will lose its properties. In the case of underestimation, function K loses the regularisation capacities and the borders of the decision area become sensitive to the noisy data. The Gaussian kernel functions belong among the so-called radial basis functions of the form

$$\mathsf{K}(\mathbf{x},\mathbf{y}) = \exp\left(- \frac{\sum_{j=1}^{n} |x_j^a - y_j^a|^b}{2\sigma^2}\right), \quad b \le 2$$

The strong point of the Gaussian kernel is that (for a correctly chosen value of the parameter σ) it filters out effectively the noisy data and the outliers.

Every Mercer kernel function can be represented as a scalar product

$$\mathsf{K}(\mathbf{x}_i,\mathbf{x}_j) = \Phi(\mathbf{x}_i)^{\mathrm{T}}\Phi(\mathbf{x}_j) \tag{2.47}$$

where $\Phi\colon X \to \mathcal{F}$ is a nonlinear mapping of the space of objects into a highly dimensional space of features $\mathcal{F}$. An important consequence of this representation is the possibility of calculating the Euclidean distance in the space $\mathcal{F}$ without knowledge of the explicit form of the function Φ. In fact,

$$\begin{aligned}\|\Phi(\mathbf{x}_i) - \Phi(\mathbf{x}_j)\|^2 &= \big(\Phi(\mathbf{x}_i) - \Phi(\mathbf{x}_j)\big)^{\mathrm{T}}\big(\Phi(\mathbf{x}_i) - \Phi(\mathbf{x}_j)\big)\\ &= \Phi(\mathbf{x}_i)^{\mathrm{T}}\Phi(\mathbf{x}_i) + \Phi(\mathbf{x}_j)^{\mathrm{T}}\Phi(\mathbf{x}_j) - 2\Phi(\mathbf{x}_i)^{\mathrm{T}}\Phi(\mathbf{x}_j)\\ &= \mathsf{K}(\mathbf{x}_i,\mathbf{x}_i) + \mathsf{K}(\mathbf{x}_j,\mathbf{x}_j) - 2\mathsf{K}(\mathbf{x}_i,\mathbf{x}_j)\end{aligned} \tag{2.48}$$

In view of the finiteness of the set X it is convenient to form a matrix K having elements $k_{ij} = \mathsf{K}(\mathbf{x}_i,\mathbf{x}_j)$. Since $k_{ij} = \Phi(\mathbf{x}_i)^{\mathrm{T}}\Phi(\mathbf{x}_j)$, then, from the formal point of view, K is a Gram matrix, see the definition B.2.3. Given this notation, we can write down the last equality in the form

$$\|\Phi(\mathbf{x}_i) - \Phi(\mathbf{x}_j)\|^2 = k_{ii} + k_{jj} - 2k_{ij} \tag{2.49}$$

The kernel functions are being used in cluster analysis in three ways, referred to through the following terms, see [124], [373]):

(a) kernelisation of the metrics,
(b) clustering in feature space $\mathcal{F}$,
(c) description via support vectors.

In the first case we look for the prototypes in the space X, but the distance between objects and prototypes is calculated in the space of features, with the use of equation (2.48). The counterpart to the criterion function (2.30) is now constituted by

$$\begin{aligned} J_1^{\Phi} &= \sum_{j=1}^{k}\sum_{i=1}^{m} u_{ij}\|\Phi(\mathbf{x}_i) - \Phi(\boldsymbol{\mu}_j)\|^2 \\ &= \sum_{j=1}^{k}\sum_{i=1}^{m} u_{ij}\Big(\mathsf{K}(\mathbf{x}_i,\mathbf{x}_i) + \mathsf{K}(\boldsymbol{\mu}_j,\boldsymbol{\mu}_j) - 2\mathsf{K}(\mathbf{x}_i,\boldsymbol{\mu}_j)\Big) \end{aligned} \tag{2.50}$$

If, in addition, $\mathsf{K}(\mathbf{x}_i,\mathbf{x}_i) = 1$, e.g. K is a Gaussian kernel, the above function simplifies to the form

$$J_1^{\Phi} = 2\sum_{j=1}^{k}\sum_{i=1}^{m} u_{ij}\Big(1 - \mathsf{K}(\mathbf{x}_i,\boldsymbol{\mu}_j)\Big) \tag{2.51}$$

In effect, in this case the function $d(\mathbf{x},\mathbf{y}) = \sqrt{1-\mathsf{K}(\mathbf{x},\mathbf{y})}$ is a distance, and if, in addition, K is a Gaussian kernel, then $d(\mathbf{x},\mathbf{y}) \to \|\mathbf{x}-\mathbf{y})\|$ when $\sigma \to \infty$.

An example of such an algorithm is considered in deeper details in Section 3.3.5.6. The idea of calculating distances in the space of features was also made use of in the kernelised and effective algorithm of hierarchical grouping, as well as in the kernelised version of the mountain algorithm[39], see also [203].

In the second case we operate with the images $\Phi(\mathbf{x}_i)$ of the objects and we look for the prototypes $\boldsymbol{\mu}_j^{\Phi}$ in the space of features. The criterion function (2.30) takes now on the form

$$J_2^{\Phi} = \sum_{j=1}^{k}\sum_{i=1}^{m} u_{ij}\|\Phi(\mathbf{x}_i) - \boldsymbol{\mu}_j^{\Phi}\|^2 \tag{2.52}$$

where $\boldsymbol{\mu}_j^{\Phi} \in \mathcal{F}$. In Section 3.1.5.5 we show how this concept is applied to the classical k-means algorithm, and in Section 3.3.5.6.2 – to the k-fuzzy-means algorithm (FCM).

Finally, the description based on the support vectors refers to the single-class variant of the support vector machine (SVM), making it possible to find in the

[39] The mountain algorithm is a fast algorithm for determining approximate locations of centroids. See R.R. Yager & D.P. Filev. Approximate clustering via the mountain method. *IEEE Trans. on Systems, Man and Cybernetics*, 24(1994),1279-1284.

space of features the sphere of minimum radius, containing *almost* all data, that is – the data with exclusion of the *outliers* [44]. By denoting the centre of the sphere with the symbol $\mathbf{v}$, and its radius with the symbol R, we obtain the constraint of the form

$$\|\Phi(\mathbf{x}_i) - \mathbf{v}\|^2 \leq R^2 + \xi_i,\ i = 1, \ldots, m \tag{2.53}$$

where ξ_i are artificial variables. More extensive treatment of this subject is presented in Section 3.5 of the book [124].

The basic characteristics of the kernel-based clustering algorithms are as follows:

(a) They enable formation and description of clusters having shapes different from spherical or ellipsoidal.
(b) They are well adapted to analysing the incomplete data and the data containing *outliers* as well as disturbances (noise).
(c) Their shortcoming consists in the necessity of estimating additional parameters, e.g. the value of σ in the case of the Gaussian kernel.

Even though the characteristic (a) sounds highly encouraging, it turns out that the classical partitional algorithms from Section 2.4 may also be applied in such situations. We deal with this subject at greater length below.

2.5.5 Cluster ensembles

Similarly as in machine learning, where the so-called families of classifiers are used in classification (see Sections 4.5 and 4.6 in [213]), in data grouping attempts are made to enhance the effectiveness of grouping by applying the families of groupings (*cluster ensembles* [179]). This kind of approach is also referred to as aggregation of clusterings or consensus partitioning, [140]. The data set X is analysed from various points of view, and the conclusions, resulting therefrom, are used in the construction of the final partition. As noted by Strehl and Ghosh [327] it is, in a way, the problem of the so-called *knowledge reuse*, with which we deal in, for instance, marketing or banking. Thus, for instance, a company disposes of various profiles, describing the behaviour of customers in terms of demographic and geographical aspects, the history of purchases done, etc. Aggregation of such descriptions allows for formulating of composite judgments, which support the design of effective trade strategies, addressed at well selected groups of customers. It is essential that in formulation of such judgments the entire analysis does not have to be repeated from scratch, but knowledge, originating from various sources is used and creatively processed.

Example 2.5.1 *Consider a simple problem, represented by the data set*[40]*, which is shown in Figure 2.10(a). Data points are here located along two spirals, of which one is situated inside the other one. The classical k-means algorithm produces the output, which is shown in Figure 2.10(b).*

[40] Information on this data set is provided in Chapter 6.

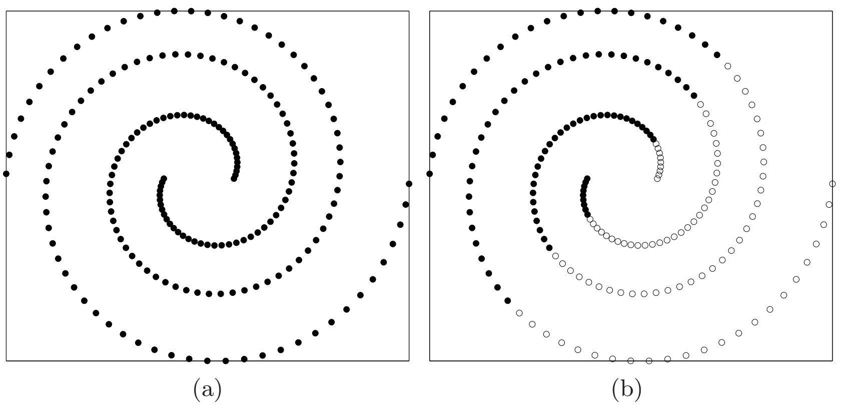

(a) (b)

Fig. 2.10. (a): The set `2spirals` is composed of two spirals, situated one inside the other. (b): grouping produced by the k-means algorithm.

In order to obtain the correct partition, Fred and Jain [132] ran N times the k-means algorithm, assuming a different number of classes at each time. They aggregated the partial results, establishing the so-called co-association matrix, composed of the elements $w_{ij} = r_{ij}/N$, where r_{ij} is the number of cases, in which the pair of objects (i, j) was assigned to the same class. In order to determine the ultimate partition on the basis of this new matrix, the single link hierarchical algorithm was used, i.e. variant (a) from Section 2.3. $\square$

The above way of proceeding can be formalised as follows: Let $\mathfrak{C} = \{\mathcal{C}^1, \ldots, \mathcal{C}^N\}$ be a family of partitions of the data set X, with $\mathcal{C}^i = \{C_1^i, \ldots, C_{k_i}^i\}$, $i = 1, \ldots, N$, where k_i is the number of groups, proposed in the i-th partition. The problem consists in finding such a partition $\mathcal{C}^*$ of the set X, which has the following properties [133]

(i) Conformity with the family of partitions $\mathfrak{C}$, i.e. the partition $\mathcal{C}^*$ ought to reflect the essential features of each partition $\mathcal{C}^i \in \mathfrak{C}$.
(ii) Robustness with respect to small disturbances in $\mathfrak{C}$, namely the number of clusters and their content ought not undergo drastic changes under the influence of slight disturbances of the partitions, forming the family $\mathfrak{C}$.
(iii) Conformity with the additional information on the elements of the set X, provided this information is available. Thus, e.g., if the assignment of (all or some) objects to classes is known, the partition $\mathcal{C}^*$ ought to be to the maximum degree in agreement with this assignment.

To measure the degree of agreement between the partitions, forming the family $\mathfrak{C}$, Fred and Jain applied in [133], similarly, anyway, as Strehl and Ghosh in [327], the normalised measure of mutual information $NMI(\mathcal{C}^\alpha, \mathcal{C}^\beta)$, where $\mathcal{C}^\alpha, \mathcal{C}^\beta$ denote the partitions compared. The form and the properties of this measure are considered in detail in Section 4.4.3. We only note here that $NMI(\mathcal{C}^\alpha, \mathcal{C}^\beta)$ is a number from the interval $[0, 1]$.

The degree of agreement of the partition $\mathcal{C}^*$ with the partitions from the family $\mathfrak{C}$ is calculated as

$$NMI(\mathcal{C}^*, \mathfrak{C}) = \frac{1}{N} \sum_{i=1}^{N} NMI(\mathcal{C}^*, \mathcal{C}^i) \tag{2.54}$$

Let, further on, $\mathfrak{P}(k) = \{\mathcal{P}_1(k), \ldots, \mathcal{P}_{\vartheta(m,k)}(k)\}$ denote all the possible partitions of the set X into k disjoint classes. It should be remembered that $\vartheta(m, k)$ is the number, defined by the equation (2.21), of all the possible partitions of an m-element set X into k disjoint classes. Hence, as $\mathcal{C}^*$ we can take the partition

$$\mathcal{C}^* = \underset{1 \leq and \leq \vartheta(m,k)}{\arg\max} \; NMI(\mathcal{P}_i(k), \mathfrak{C}) \tag{2.55}$$

This partition satisfies the first of the postulates, formulated before. In order to account for the requirements of flexibility, the authors quoted here form with a bootstrap method an M-element family of partitions $\mathbb{B} = \{\mathfrak{B}_1, \ldots, \mathfrak{B}_M\}$, randomly assigning objects from the set X, with repetitions, to the appropriate sets from the family $\mathfrak{C}$. A more extensive treatment of the subject, along with a description of the performed experiments, is provided in [133] and [134].

Alternative methods of aggregating multiple partitions of the data set are considered by Strehl and Ghosh [327]. Hore, Hall and Goldgof [179] formulate the procedure that ensures scalability of aggregation of partitions (represented by the gravity centres of classes), corresponding to a sparse data set. These latter authors consider two situations: (a) in each portion of data the same number of classes is distinguished, or (b) the numbers of classes are different. In the first case these authors obtain the so-called BM (*Bipartire Merger*) algorithm, and in the second case – the MM (*Metis Merger*) algorithm. An additional strong point of this work is the rich bibliography, concerning the families of partitions.

Then, Thangavel and Visalakshi [333] describe the application of the families of partitions in the k-harmonic means algorithm[41]. Finally, Kuncheva and Vetrov wonder in [222] whether the outputs from cluster ensembles are more stable than partitions obtained from the a single clustering algorithm. They understand stability as sensitivity (or, more precisely, lack of sensitivity) with respect to small disturbances in the data or in the parameters of the grouping algorithm. The considerations therein allowed for the formulation of certain recommendations, concerning the selection of the number of clusters, resulting from the analysed family of partitions.

Let us mention, at the end, one more method of aggregating the partial groupings, which is used in microarray analysis.

Example 2.5.2 *One of the most popular applications of clustering in bioinformatics is microarray analysis. Suppose we treat X as a matrix with rows*

[41] This algorithm is presented in Section 3.1.5.4 of this book.

corresponding to genes, and columns – to experiments or samples. The value of x_{ij} *corresponds to the level of expression of the i-th gene in the sample (experiment) j. In typical applications from this domain the matrix X is exceptionally "slender": it has thousands of rows and not more than 100 columns.*

The study [63] presents the problem of formation of meta-genes, being the linear combinations of n genes. In solving this problem, the non-negative factorisation of matrices is used, mentioned here in Section 2.4.2.1, i.e. finding of such matrices W and H, whose product WH^T *is an approximation of the matrix X. In this concrete case columns of the matrix W correspond to meta-genes (or to diagnostic classes), while the number* w_{ij} *defines the value of the coefficient of contribution from the i-th gene in the j-th meta-gene. Then, the elements* h_{ij} *of the matrix H indicate the levels of expression of the meta-gene j in sample i. Matrix H is made use of for grouping of samples: i-th sample is assigned to this meta-gene* j^**, which corresponds to the maximum value of* h_{ij}*.*

In the general case, by performing the decomposition of the matrix X many times over, we obtain the set of matrices (W^t, H^t)*, where* $t = 1, \dots, t_{max}$ *denotes the successive number of the NMF decomposition, while* t_{max} *is the total number of the decompositions performed.*

In the study, reported in [63], the following manner of aggregating the partial results was applied. Let C^t *denote the concordance matrix, obtained in experiment t, having dimensions* $m \times m$*. Its element* $c^t_{ij} = 1$ *if genes i and j belong to the same class, and* $c^t_{ij} = 0$ *in the opposite case. Let, further,* $\overline{C} = (C_1 + \cdots + C_{t_{max}})/t_{max}$ *be the aggregate (averaged) concordance matrix. The numbers* $\overline{c}_{ij}$ *can be treated as the degrees of similarity of the gene pairs* (i, j)*. By turning similarities into distances, that is - by forming the matrix D, having elements* $d_{ij} = 1 - \overline{c}_{ij}$*, we construct the dendrogram and calculate the cophenetic correlation coefficient (see example 2.3.1 in page 38), indicating the degree of agreement between the distances, contained in matrix D, and the distances, resulting from the dendrogram developed. If the results of grouping, obtained in each run of the algorithm are similar (meaning that the grouping obtained has a stable character), then the elements of matrix* $\overline{C}$ *(and of matrix D) will have values close to 0 or 1, and the calculated correlation coefficient will be close to 1. In case, when the data set analysed does not represent a clear k-group structure, the correlation coefficient shall have value well below 1. In addition, the resulting dendrogram serves in the ordering of column and rows of the concordance matrix. The use is made here of the order, in which the leaves of the dendrogram are marked. In the left hand part of Figure 2.11 the unordered matrices of concordance,* $\overline{C}$*, are shown, as obtained for* $k = 2, 3, 4, 5$*, while in the right-had part – the same matrices with the appropriately ordered rows and columns. Matrix X represents, in this case, the levels of expression of 5000 genes, registered in 38 samples, taken from the bone marrow* [42]*. Conform to the claim from the authors of [63], the method here outlined allows for a precise distinction between*

[42] The data, as well as the MATLAB code, are available at `http://www.broadinstitute.org/cgi-bin/cancer/publications/pub_paper.cgi?mode=view&paper_id=89`.

two types of leukaemia (myeloid and lymphoblastic leukaemia), this being indicated in the upper matrix in part (b) of Figure 2.11. A Reader, interested in the interpretation of the remaining figures is kindly referred to the publication [63].

It should be noted that the quality of partitions, as measured by the cophenetic correlation coefficient, decreases with the number of groups – see Figure 2.12. This corresponds to the existence of less distinct structures in the data set, so that the algorithm is not capable of determining them sufficiently precisely. □

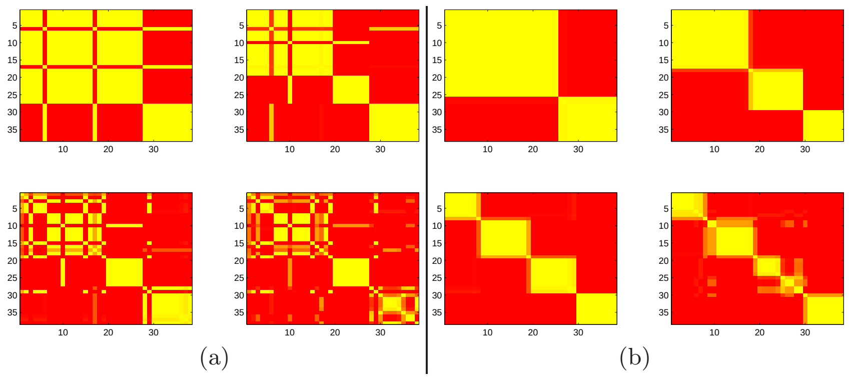


Fig. 2.11. Unordered (a) and ordered (b) concordance matrices, generated with the use of the method, described in the text. Experiments were carried out for $k = 2, 3, 4, 5$ groups

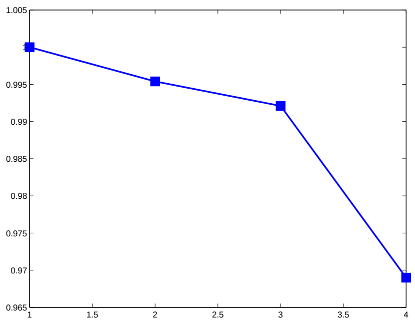


Fig. 2.12. Values of the cophenetic correlation coefficient for the partitions obtained. The horizontal axis represents the number of groups, the vertical axis shows the cophenetic correlation coefficient.

2.6 Whether and when grouping is difficult?

When treated as an optimisation task, grouping is a "hard" problem, if we consider it in the context of pessimistic complexity (the worst case analysis). This means that with the increase of the number of observations there is a dramatic increase in the pessimistic time complexity, associated with finding the global optimum of the criterion function. If, however, the data represent the true clusters, and the number of observations is sufficiently high, then use can be made of a number of methods of local search, allowing for the correct identification of groups. This may lead to the conviction that "grouping is not difficult; it is either easy or not interesting" [324].

It turns out, in fact, that if the data originate from a (well separable) mixture of normal distributions, and the number of observations is sufficiently high, then the task of grouping is easy. There exists an algorithm, having polynomial time complexity, which identifies – with high probability – the correct division into groups. In particular, this algorithm locates sufficiently precisely (with an assumed error) the gravity centres of groups. This allows for formulating the upper bound on the computational conditioning of the grouping algorithm, that is – the minimum gap between groups and the minimum number of observations in the sample, ensuring correct classification. So, for instance, for an arbitrary Gaussian mixture, if only an appropriate number of observations exist, then the maximum likelihood estimates tend to the true parameters, provided that the local maxima of the likelihood function are lower than the global maximum [293].

One can, of course – and, in fact, should – ask what the "sufficiently high" number of observations means, that is – what is the information limit for the task of grouping. This issue is discussed also in the already cited work [324].

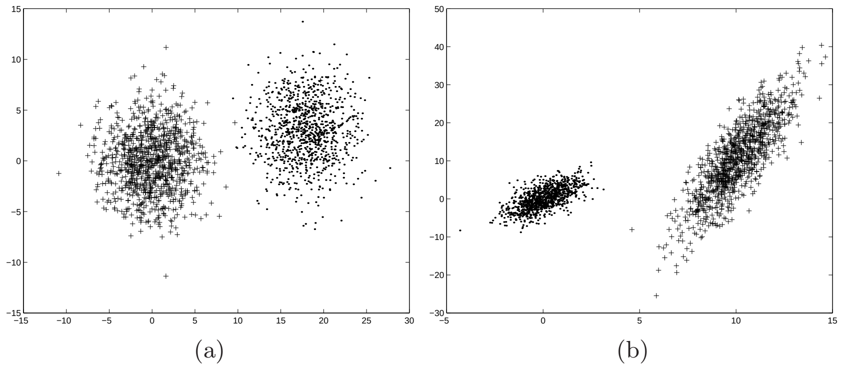


Fig. 2.13. Exemplary data having: (a) spherical and (b) non-spherical normal distribution.

Dasgupta and Schulman [87] concentrate on the mixture of n-dimensional spherical normal distributions $N(\boldsymbol{\mu}, \sigma\mathbb{I}_n)$ – see Figure 2.13. The data, which come from a spherical normal distribution, can be enclosed inside the hypersphere with the radius[43] $r = \sigma\sqrt{n}$. Therefore, it can be assumed that the data, originating from two distributions, $N(\boldsymbol{\mu}_1, \sigma_1\mathbb{I}_n)$ and $N(\boldsymbol{\mu}_2, \sigma_2\mathbb{I}_n)$ are c-separable, if [87]

$$\|\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2\| \geq c \max(\sigma_1, \sigma_2)\sqrt{n} = c \max(r_1, r_2) \tag{2.56}$$

We have endowed, in this manner, the notion of separability of distributions, with a precise meaning. In particular, when we deal with anisotropic multinormal distributions $N(\boldsymbol{\mu}, \Sigma)$, where Σ denotes the covariance matrix, then $r = \sqrt{tr(\Sigma)}$. In the case of a mixture of k distributions, we denote by c_{ij} the separability of the i-th and j-th distribution, and by $c = \min_{i \neq j} c_{ij}$ the separability of the mixture. So, e.g., the 2-separable mixture of distributions represents almost exclusively the disjoint clusters of n dimensional points, and with the increasing n lower and lower value of c is required to secure the disjointness of clusters, this being due to the specific properties of the Euclidean distance (see Section 2.2).

If the mixture of k spherical normal distributions is sufficiently separable (at the order of $\Omega(n^{1/4})$), and the sample contains $O(k)$ observations, then it suffices to perform two iterations of the EM algorithm[44]. Further advance is achieved by the application of the spectral projection methods (considered in Chapter 5). Vempala and Wang [348] show that if c is a constant of the order $\Omega(n^{1/4} \log^{1/4} nk)$ and the dimension of the sample is of the order $\Omega(n^3 k^2 \log ckn/\delta)$, then the k-dimensional spectral projection allows for identification of the group centres with probability $1 - \delta$.

Kanungo *et al.* adopt, as the measure of separation of clusters, in [197], the quotient

$$sep = \frac{r_{min}}{\sigma_{max}} \tag{2.57}$$

where r_{min} is half of distance between the closest centres of classes, while σ_{max} denotes the maximum of the standard deviations, characterising clusters. The authors quoted show, see Theorem 1 in [197], that if the class centres are sufficiently close to the gravity centres of clusters, then as the value of *sep* increases, the time of execution of the appropriately implemented k-means algorithm improves.

A similar conclusion was formulated by Zhang in the report [384]. Call *clusterability* a measure characterising the partition C of the set X. For a given partition $\mathcal{C} = \{X_1, \ldots, X_k\}$ it is, for instance, possible to define the within-group variance, $W_C(X) = \sum_{j=1}^k p_i \sigma^2(X_i)$, and the between-group variance, $B_C(X) = \sum_{j=1}^k (\boldsymbol{\mu}_j - \boldsymbol{\mu})$, where $p_i = |X_i|/m$, $\boldsymbol{\mu}_j$ is the gravity centre of the

[43] It is, actually, the approximate average length of the random vector, having exactly this distribution. If $\mathbf{x}$ is a vector, having as coordinates random numbers distributed according to $N(0, \sigma)$, then its expected length is $\mathbb{E}(\|\mathbf{x}\|) = \sigma\sqrt{2}\Gamma((n+1)/2)/\Gamma(n/2)$. In an approximation, $\mathbb{E}(\|\mathbf{x}\|) \approx \sigma\sqrt{2}[1 - 1/(4n) + 1/(21n^2)]$, and so $\mathbb{E}(\|\mathbf{x}\|) \to \sigma\sqrt{2}$ when $n \to \infty$.

[44] This algorithm is commented upon in Section 3.2.

j-th group, and $\boldsymbol{\mu}$ is the gravity centre of the entire set X. The, the measure of clusterability is the quotient [384]

$$C(X) = \max_{C \in \mathcal{C}} \frac{B_C(X)}{W_C(X)}$$

The higher the value of $C(X)$, the more separate are individual groups. One of the results, presented in the report quoted, proposes that the higher the clusterability (corresponding to the existence of natural clusters), the easier it is to find the appropriate partition [2].

Let us terminate this section with the following statement. The assessment of quality of a concrete tool, in this case – an algorithm of grouping – remains, actually, in the competence of the person, using the tool. Thus, instead of looking for the "best" tool, one should rather consider the available algorithms in the categories of their *complementarity*: the capacity of compensating for the weak points of one algorithm with the qualities of the other, or the capacity of strengthening the positive qualities of an algorithm by some other one.

3 Algorithms of combinatorial cluster analysis

Let us have a look at basic algorithms of cluster construction via target function optimisation.

3.1 *k*-means algorithm

k-means is considered to be the most typical representative of this group of algorithms. Early outlines of this algorithm can be found in papers of Steinhaus [326] (published in 1956), Lloyd [238] (1957), Ball and Hall [35] (1965) and MacQueena [240] (1967). The authors of [368] place it among the ten most important data mining algorithms. The 50-years-long history of the algorithm is exposed by Jain in [190].

As a matter of fact, the modern version on the k-means algorithm is an adaptation of Lloyd heuristic[1], [238]. Originally, the heuristic was proposed for scalar quantisation in PCM (*Pulse Code Modulation*) systems. Its essence lies in alternating assignment of objects to clusters (based on known centroid location) and prototype updates (based on known cluster membership of objects). It is assumed that the signal, subject to quantisation, is in general a random vector X with a probability density function f_X. Actually, the density function f_X is not known and only a sample $X = \{\mathbf{x}_1, \dots, \mathbf{x}_m\}$ is available. An adaptation of Lloyd algorithm to such conditions is called LBG algorithm [235], which is nearly identical with the k-means algorithm presented below. Like any other heuristic, this algorithm does not guarantee finding the optimal solution. Yet, it was observed that the algorithm behaves very well if the elements of the data set form natural groups with different properties. The more the properties of these groups differ, the fewer iterations are needed to discover the internal structure of the data set[2].

In spite of more than half century that passed since invention of the k-means algorithm, its properties and convergence conditions have not been sufficiently investigated. A reader interested in these aspects is advised to have a look at

[1] Though the algorithm was presented in a report dated July 31st, 1957, it was not officially published till 1982. A similar algorithm was published earlier in a paper by J. Max, Quantizing for minimum distortion. *IEEE Trans. on Info. Th.*, **50**(5),937-994, 1960. For this reason it is called also Lloyd-Max algorithm, or Max-Lloyd algorithm.

[2] We mentioned this property in Section 2.6.

papers[3] [273], [22] and the bibliography contained therein. We recommend also the paper [21], the authors of which exploit the so-called *smoothed analysis*[4].

In case of k-means algorithm the target is assumed to be the minimisation of the trace of the matrix W, defined by equation (2.27). Let U denote once more the *assignment matrix* , telling which object belongs to which cluster (cluster assignment to or split/partition into clusters), and M – the matrix with rows representing gravity centres of the clusters[5]. Then the optimisation criterion function, called also the partition cost function, is of the form

$$J(U,M) = \sum_{i=1}^{m}\sum_{j=1}^{k} u_{ij}\|\mathbf{x}_i - \boldsymbol{\mu}_j\|^2 \tag{3.1}$$

Its minimisation is equivalent to minimisation of the sum of error squares. Here, an "error" is understood as the distance of the i-th observation from the centre of the cluster to which it belongs.

In general, the Euclidean distance $\|\mathbf{x}_i - \boldsymbol{\mu}_j\|$ can be replaced by the Minkowski distance $d_p(\mathbf{x}_i - \boldsymbol{\mu}_j)$ and the criterion function (3.1) will then have the form

$$J_p(U,M) = \sum_{i=1}^{m}\sum_{j=1}^{k} u_{ij} d_p^p(\mathbf{x}_i, \boldsymbol{\mu}_j) = \sum_{i=1}^{m}\sum_{j=1}^{k} u_{ij}|\mathbf{x}_i - \boldsymbol{\mu}_j|^p \tag{3.2}$$

The points, at space in which the partial derivatives [6] are equal zero, are good candidates for minima of the partition cost function:

$$\begin{aligned}\frac{\partial}{\partial\boldsymbol{\mu}_t} J_p(U,M) &= \sum_{i=1}^{m}\sum_{j=1}^{k} u_{ij}\frac{\partial}{\partial\boldsymbol{\mu}_t}|\mathbf{x}_i - \boldsymbol{\mu}_j|^p \\ &= \sum_{i=1}^{m} u_{it}\frac{\partial}{\partial\boldsymbol{\mu}_t}|\mathbf{x}_i - \boldsymbol{\mu}_t|^p \\ &= \sum_{\mathbf{x}_i\in C_t} p|\mathbf{x}_i - \boldsymbol{\mu}_t|^{p-1}\end{aligned}$$

[3] See also Q. Du, M. Emelianenko, and L. Ju, Convergence of the Lloyd algorithm for computing centroidal Voronoi tessellations, *SIAM J. on Numerical Analysis*, **44**(1), 102–119, 2006; M. Emelianenko, L. Ju, and A. Rand, Nondegeneracy and weak global convergence of the Lloyd algorithm in $\mathbb{R}^d$, *SIAM J. on Numerical Analysis*, **46**(3), 1423–1441, 2009

[4] It is a mixture (a hybrid) of worst-case and average-case analyses. It allows to estimate realistically the algorithm complexity in practical settings. The research in this area was initiated by D. Spielman and S.-H. Teng with the paper "Smoothed analysis of algorithms: why the simplex algorithm usually takes polynomial time", *Proc. of the 33-rd Annual ACM Symposium on Theory of Computing*, ACM 2001, pp. 296–305. We recommend "Mini course on smoothed analysis" available at the Web site `http://roeglin.org/publications/SmoothedAnalysis.pdf`.

[5] cluster gravity centres are called also cluster prototypes, or class centres or just prototypes

[6] We treat here J_p as a *continuous* function of coordinates of cluster centre coordinates.

Components of the gradient vector are of the form

$$\frac{\partial}{\partial \mu_{tl}} |x_{il} - \mu_{tl}|^p = p \cdot \mathtt{sgn}(x_{il} - \mu_{tl}) \cdot |x_{il} - \mu_{tl}|^{p-1}$$

In particular, when $p = 2$, we get

$$\sum_{\mathbf{x}_i \in C_t} 2(x_{il} - \mu_{tl}) = 0 \Rightarrow \mu_{tl} = \frac{1}{n_t} \sum_{\mathbf{x}_i \in C_t} x_{il} \tag{3.3}$$

where n_t means the cardinality of the class C_t.

It must be stressed, however, that, for a fixed set X with cardinality m, the cost of partitioning this set into k clusters via gravity centres determined by the above equation is a function which: (a) takes on at most $\vartheta(m, k)$ distinct values[7] and (b) possesses many local minima. For example, in Figure 3.1, a diagram is shown of the partition cost as a function of gravity centre of *one* of the clusters.[8] The set X consists here of 10 points. Five of them were drawn randomly according to the rule[9] $3 + \mathtt{rand}$, and the remaining five – according to the rule $8 + \mathtt{rand}$.

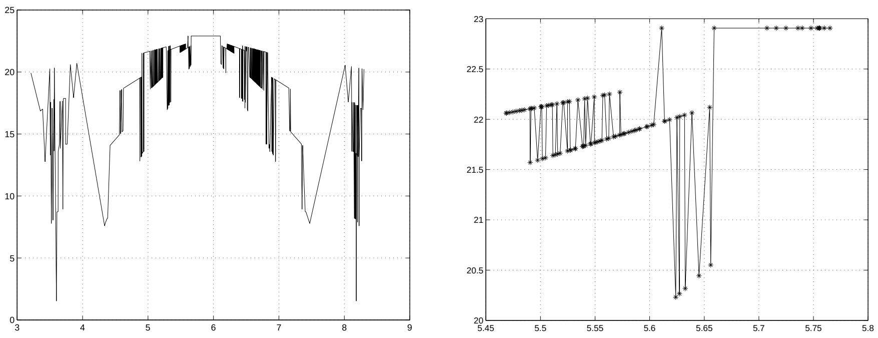


Fig. 3.1. Local optima in the problem of partitioning of the (one dimensional) set X, consisting of 10 points. Figure to the left: values of the indicator J (vertical axis) for the gravity centre of one cluster placed at locations from the interval $[3, 9]$. Figure to the right: a detailed look at these values when the centre belongs to the set $[5.4686, 5.7651]$.

The above remarks indicate that the task of determining an optimal split is $\mathcal{NP}$-hard. It is that hard even if $k = 2$, and also even when $n = 2$ (see [20, p. 4] and the bibliography cited therein). This fact justifies application of

[7] The number $\vartheta(m, k)$ is defined in page 39.

[8] Note that the gravity centre of the second cluster would make the diagram more complex

[9] $\mathtt{rand}$ means a random number generator sampling with uniform distribution from the interval $[0, 1]$.

various heuristics [10]. However, a common drawback of the heuristics is the lack of warranties on achieving an optimal solution. For example, in case of the Lloyd heuristic, that is, the variant of the algorithm 2.2 with weights $w(\mathbf{x}_i) \equiv 1$, $i = 1, \dots, m$, the ratio of the cost returned by the algorithm to the optimal cost can be any large number. It is not hard to encounter such situations, as illustrated by the example below:

Example 3.1.1 *[20, p. 89] Assume that the set $\mathfrak{X}$ consists of four points with coordinates: $\mathbf{x}_1 = (-a, -1)$, $\mathbf{x}_2 = (a, -1)$, $\mathbf{x}_3 = (-a, 1)$ and $\mathbf{x}_4 = (a, 1)$, where $a \geq 1$. If $k = 2$, and initial cluster centres are located at points $\mathbf{x}_1$ and $\mathbf{x}_3$, then the k-means algorithm returns clusters $C_1 = \{\boldsymbol{x}_1, \boldsymbol{x}_2\}$, $C_2 = \{\boldsymbol{x}_3, \boldsymbol{x}_4\}$ with centres $\boldsymbol{\mu}_1 = (0, -1)$, $\boldsymbol{\mu}_2 = (0, 1)$ and with cost $J = 4a^2$. But if the initial centres were placed at points $\mathbf{x}_1$ i $\mathbf{x}_4$ then we get the optimal split $C_1' = \{\boldsymbol{x}_1, \boldsymbol{x}_3\}$, $C_2' = \{\boldsymbol{x}_2, \boldsymbol{x}_4\}$ with cost $J_{opt} = 4$ and centres $\boldsymbol{\mu}_1' = (-a, 0)$, $\boldsymbol{\mu}_2' = (a, 0)$. This means that $J/J_{opt} = a^2 \to \infty$ when $a \to \infty$.*

Let us note, in passing, that for $a < 1$ the optimal split is of the form $C_1 = \{\boldsymbol{x}_1, \boldsymbol{x}_2\}$, $C_2 = \{\boldsymbol{x}_3, \boldsymbol{x}_4\}$. Also in this case the quotient $J/J_{opt} = 1/a^2 \to \infty$ when $a \to 0$. □

For this reason there emerges a growing interest in approximation algorithms, that is, algorithms returning a partition with cost $J = (1 + \varepsilon)J_{opt}$, as mentioned by Arthur in page 4 of his thesis [20]. Regrettably, in the majority of cases these algorithms have exponential complexity in the number of clusters and the amount of data. This fact renders them next to useless in the analysis of big data sets. An exception here is constituted by the papers [196] and [219].

Subsequently, we describe the base variant of the k-means algorithm and a number of its modifications.

3.1.1 The batch variant of the *k*-means algorithm

The control flow of the base algorithm is depicted in pseudo-code 2.2 on page 48. It is assumed that $w(\mathbf{x}_i) = 1, i = 1, \dots, m$. The algorithm starts with determination of gravity centres of the clusters. The simplest, though least efficient, initialisation method, namely a random split of the set of objects into k clusters and computation of the mean value in each cluster, is applied. Then, repeatedly, two steps are executed: (a) each object is moved to the cluster with the closest centre to it, and subsequently (b) in each cluster its centre is updated given the new object assignment. The procedure is terminated when the cluster assignment stabilizes [11].

[10] See Hruschka, E.R., *et al.* A survey of evolutionary algorithms for clustering. *IEEE Trans. on Systems Man, and Cybernetics, Part C: Applications and Reviews*, **39**(2), 2009, 133-155; A. Ajith, S. Das, S. Roy. Swarm intelligence algorithms for data clustering. *Soft Computing for Knowledge Discovery and Data Mining.* Springer US, 2008. 279-313.

[11] If U^t denotes the split matrix obtained in the t-th iteration, then the split corresponding to it is called stabilised one if $U^{t+1} = U^t$.

Such a *batch* version of the algorithm is called also Forgy algorithm [128] or H-MEANS heuristic [319]. It resembles the EM algorithm, described in Section 3.2. Selim and Ismail have shown in [310] that the algorithm converges in a finite number of steps and have defined conditions for which the obtained solution is a local minimum of the function (3.1).

3.1.2 The incremental variant of the *k*-means algorithm

The "classic version" (from the point of view of code optimisation) of the algorithm passes iteratively through *each* object and checks if its relocation to some other cluster would improve the target function. In case such an improvement is observed, the object changes its cluster membership. Let us assume we want to move a certain object $\mathbf{x}^*$ from cluster C_j to cluster C_l. Then, the new coordinates of the gravity centre of this latter cluster would be computed from the formula [112]:

$$\boldsymbol{\mu}_l^* = \frac{n_l \cdot \boldsymbol{\mu}_l + \mathbf{x}^*}{n_l + 1} = \boldsymbol{\mu}_l + \frac{\mathbf{x}^* - \boldsymbol{\mu}_l}{n_l + 1}$$

The coordinates of the gravity centre of the cluster C_j, after removal of the object $\mathbf{x}^*$ from it, undergo a similar transformation:

$$\boldsymbol{\mu}_j^* = \boldsymbol{\mu}_j - \frac{\mathbf{x}^* - \boldsymbol{\mu}_j}{n_j - 1}$$

Let us denote by $V(C_j)$ the sum of squares of distances of all objects of the cluster C_j from its gravity centre. It can be easily verified that (see also [112]):

$$V(C_j - \{\mathbf{x}^*\}) = V(C_j) - \frac{n_j}{n_j - 1}\|\mathbf{x}^* - \boldsymbol{\mu}_j\|^2 = V(C_j) - \delta_j(\mathbf{x}^*)$$

$$V(C_l \cup \{\mathbf{x}^*\}) = V(C_l) + \frac{n_l}{n_l + 1}\|\mathbf{x}^* - \boldsymbol{\mu}_l\|^2 = V(C_l) + \delta_l(\mathbf{x}^*)$$

So, we see that the relocation of the object $\mathbf{x}^*$ would be advantageous if the condition $\delta_j(\mathbf{x}^*) > \delta_l(\mathbf{x}^*)$ holds, that is:

$$\frac{n_j}{n_j - 1}\|\mathbf{x}^* - \boldsymbol{\mu}_j\|^2 > \frac{n_l}{n_l + 1}\|\mathbf{x}^* - \boldsymbol{\mu}_l\|^2$$

We obtained in this way an algorithm called in [112] BIMSEC (*Basic Iterative Minimum-Squared-Error Clustering*) or K-MEANS (see e.g. [319]). Pseudo-code 3.1 describes the basic steps of the algorithm.

Though the "classic" incremental version is more expensive than the "batch" version (H-MEANS algorithm), it turns out to be more successful [112]. One can say that while a solution returned by K-MEANS cannot be improved by H-MEANS, it is nonetheless possible to improve the result of H-MEANS by subsequent application of K-MEANS – see, e.g., [163]. A compromise between computational efficiency and accuracy can be achieved via alternating application of both algorithms. For example, we may repeat in a loop many times the steps (2) and (3) of the

Algorithm 3.1 Incremental clustering algorithm BIMSEC

```
1: Determine the initial split of the data set into $k$ non-empty clusters and compute
   their gravity centres.
2: Choose a (next) element $\mathbf{x}^*$ from a cluster, say cluster $C_j$.
3: If $n_j = 1$ – go to step 6. Otherwise compute the increments
   $$\delta_l(\mathbf{x}^*) = \begin{cases} \frac{n_l}{n_l+1}\|\mathbf{x}^* - \boldsymbol{\mu}_l\|^2 & \text{if } l \neq j \\ \frac{n_j}{n_j-1}\|\mathbf{x}^* - \boldsymbol{\mu}_j\| & \text{if } l = j \end{cases}$$
4: Move object $\mathbf{x}^*$ to the cluster $C_{j^*}$ if $\delta_{j^*} \le \delta_l, l = 1, \dots, k$.
5: Compute new coordinates of gravity centres $\boldsymbol{\mu}_j, \boldsymbol{\mu}_{j^*}$ and the value of the indicator
   $J(U, M)$.
6: If the indicator value did not change after testing $m$ objects – halt. Otherwise go
   back to step 2.
```

algorithm 2.2 and thereafter only several times (due to higher computational costs) we would carry out object moves according to the `K-MEANS` heuristic.

3.1.3 Initialisation methods for the *k*-means algorithm

It is beyond doubt that the big advantage of the k-means algorithm for big data analysis is its simplicity and linear time complexity with respect to the number of objects. It has, however, also a number of disadvantages, the most important of them being the following:

(a) The result depends on the order, in which the data is processed and on the value range (scale) of individual components of the vector, describing the objects. Initial normalisation allows to eliminate the impact of value ranges.
(b) It is a greedy algorithm the result of which depends on initial conditions, see Figure 3.2.
(c) It is sensitive to the presence of abnormal observations (*outliers*).
(d) The number of clusters must be known in advance.
(e) It can be used only for analysis of numerical data. Simple generalisations for the case of mixed (qualitative and quantitative) data are discussed in Section 3.1.5.7.

Furthermore, application of the `H-MEANS` heuristic leads frequently to obtaining several empty clusters [319]. In such cases the `H-MEANS+` heuristic, introduced by Hansen and Mladenovic in [163], may prove helpful. If one got $k - k_1$ non-degenerate (i.e. non-empty) clusters, then k_1 objects are selected that are most distant from the centres of clusters, to which they belong. These objects are treated as new, additional cluster centres and all the objects are re-classified according to the rule "the winner takes all", (2.43).

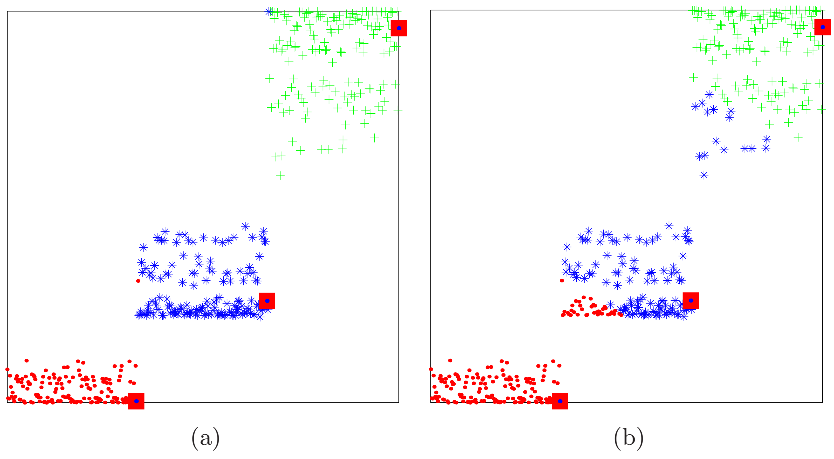


Fig. 3.2. The influence of initialisation and scaling on the results of clustering of a set of 450 observations. Initial location of cluster centres is marked with red squares, and respective symbols denote the assignment of objects to clusters. In case (a) $x_{i1} \in \{1, \dots, 450\}$, and in case (b) $x_{i1} \in 2.222185{\cdot}10^{-3} + 2.034705{\cdot}10^{-10} \cdot \{1, \dots, 450\}$, $i = 1, \dots, 450$

In order to reduce the impact of initialisation on the final result, a number of "clever" initialisation methods were proposed. Let us present a couple of the most frequently used methods.

(a) k objects randomly selected from the set X are treated as cluster centres. The remaining objects are assigned to respective clusters according to the clustering update rule, that is, they are assigned to the closest gravity centre. This method was proposed by Forgy in [128].
(b) As previously, we start with a random choice of k gravity centres and an order is imposed on the set of objects. The objects (visited according to the imposed order) are assigned to the cluster with the closest gravity centre and after this assignment the coordinates of gravity centre of this cluster are updated. This method was suggested by MacQueen in 1967.
(c) The next method was proposed by many authors. Among the first ones we shall mention Goznalez [145] and Hochbaum and Shmoys [176]; it was also suggested, in particular, by Katsavounidis, Kuo and Zhang [200]. The crucial idea of the method is to choose from the data set k objects most distant one from another. In the first step, the $\boldsymbol{\mu}_1$ is set to the coordinates of the object $\mathbf{x} \in X$ of maximal length, that is $\boldsymbol{\mu}^1 = \arg\max_{1 \le i \le m} \|\mathbf{x}_i\|$. It is the seed of the set of centroids M, that is: $M = \{\boldsymbol{\mu}^1\}$. Subsequently, the object $\mathbf{x} \in X$ is identified that is most distant from $\boldsymbol{\mu}^1$. This is the second centroid $\boldsymbol{\mu}_2$ inserted into the set M. To determine the j-th candidate, for each $\mathbf{x} \in X \backslash M$ the distance from all elements from the set M is computed and the smallest one is assumed to be the distance $d(\mathbf{x}, M)$. The object most distant from the

set M is selected as $\boldsymbol{\mu}_j$, that is

$$\boldsymbol{\mu}_j = \arg\max_{\mathbf{x}_i \in X} \Big(\min_{\boldsymbol{\mu} \in M} \|\mathbf{x}_i - \boldsymbol{\mu}\| \Big) \tag{3.4}$$

The process of choosing the object most distant from M is terminated, when we get k candidates. A quick algorithm, implementing this method, is presented in Section 3.1.4, and its probabilistic variant – in Section 3.1.3.1.

(d) Ng, Jordan and Weiss proposed in [269] to choose the initial centroids in such a way that the vectors representing them are as orthogonal to one another as possible. Elements $\mathbf{x}^j$ of the set M are selected from the set X in such a way that the vectors representing them obey the above mentioned condition. The technique to choose them properly is as follows: The element $\mathbf{x}^1$ is picked at random from X. If M is a matrix with dimensions $j \times n$, and X a matrix with dimension $m \times n$, then the elements of the product $\cos = XM^{\mathrm{T}}$ represent the cosine of the angle between the individual objects and the current set of gravity centres (given, of course, that the respective vectors are normalized). For each object, the maximal absolute value is selected, $\cos^*_i = \max_{1 \le l \le j} |\cos_{il}|$, and the next centroid is the object for which this cosine is the lowest. The procedure is repeated for $j = 2, \ldots, k$. As mentioned, an introductory normalization of the rows of the matrix X (so that each row vector is of length 1) is necessary to obtain the correct results.

(e) Another, most elaborate method, is co-authored by Kaufman and Rousseeuw, [201]. The cluster centres are chosen iteratively till we get k of them. The first centroid $\boldsymbol{\mu}_1$ is the most central object of the whole data set. Assume we chose already $s < k$ gravity centres. For each pair of objects, $\mathbf{x}_i, \mathbf{x}_j$, not selected into the set of centroids, the value $\beta_{ij} = \max[B_j - d(\mathbf{x}_i, \mathbf{x}_j), 0]$ is computed, where $B_j = \min_{l=1,\ldots,s} d(\mathbf{x}_j, \boldsymbol{\mu}_l)$. Thereafter, the gain from selecting the location $\mathbf{x}_i$ equal $\sum_j \beta_{ij}$, is computed and the location $\mathbf{x}_i$, is selected as the next gravity centre, which maximizes this gain value.

(f) An initial clustering of the set of objects is obtained using some other method, like the Ward algorithm.

Experiments described in [281] indicate quite good properties of random cluster initialisation and of Kaufman/Rousseeuw initialisation. Ng *et al.* claim that their initialisation method ensures a quick convergence of the clustering algorithm, [269]. Other initialisation methods were elaborated by Su and Dy in [328], though their proposals, we think, seem to be quite complex algorithmically. A number of bibliographic positions devoted to this topic are mentioned by Kuncheva and Vetrow [222]. A careful comparison of various initialisation methods can be found in [244].

Figure 3.3 illustrates the qualitative difference between methods (c) and (d). Note that in both cases each candidate centre belongs to a different cluster.

Still another idea is the random initialisation of the assignment matrix. However, it is a much worse solution. For example, for the data set `data3_2.txt`, in the case of random initialisation of the class centres, we got 72 correct solutions

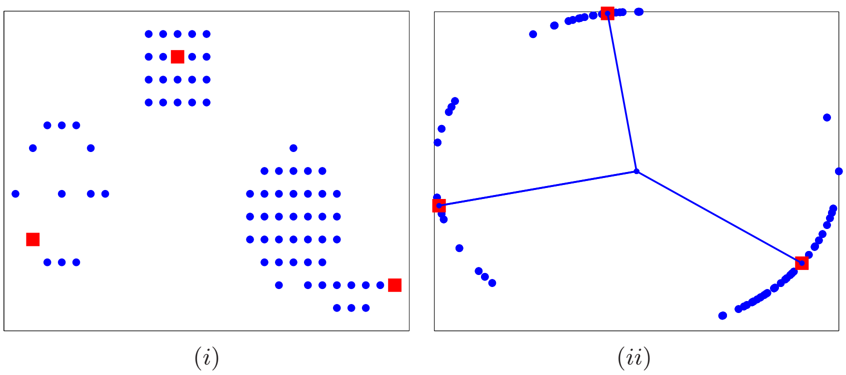


Fig. 3.3. The distribution of the gravity centres (denoted with squares) in two methods of initialisation. (i) – initialisation described in point (c), (ii) – initialisation described in point (d). In this last case the coordinates were initially normalised in such a way that $\mathbf{x}_{ij} \in [-1, 1]$, $i = 1, \dots, m$, $j = 1, \dots, n$, and afterwards – as required by the method – each row of the matrix X was so normalized that $\|\mathbf{x}_i\| = 1$. The lines connecting the coordinate system origin $(0, 0)$ with gravity centres play here an explanatory role only – they allow to imagine the angles between the centres

in 100 runs, while in the case of random initialisation of the U matrix – only 38 correct solutions in 100 runs.

A thorough analysis of the impact of initialisation on the stability of k-means algorithm was carried out in [64]. An advanced initialisation method was also proposed there. It is a two-stage procedure. In the first stage, $k' > k$ gravity centres are picked randomly, objects are assigned to corresponding clusters and the centre coordinates are updated. Thereafter, most "valuable" centres are selected, and among them, using the variant (c) – k centres most distant from one another are chosen.

3.1.3.1 k-means++ algorithm

An interesting initialisation method was presented in [22] (compare also [273]). It is, in fact, a variant of the (c) method from the previous section. Its authors did notice that if the data set contains at least k outliers, then the choice of the most distant observations will result in malformed initial centres. Hence, they propose a probabilistic variant of the selection rule. Let $u(\mathbf{x})$ denote the distance of the object $\mathbf{x}$ from the set C consisting of $j < k$ centres. The next centre is picked according to the probability distribution[12]

$$p(\mathbf{x}) = \frac{u^2(\mathbf{x})}{\sum_{\mathbf{x}' \in X} u^2(\mathbf{x}')} \tag{3.5}$$

[12] If we replace the Euclidean distance with Minkowski d_l then the elements of the set X should be picked with probability $p(\mathbf{x}) = u^l(\mathbf{x}) / \sum_{\mathbf{x}' \in X} u^l(\mathbf{x}')$. Consult [20, p. 98].

The pseudo-code 3.2 describes the steps of the algorithm. The procedure of probabilistic selection of candidate centres is in fact very quick. One can apply the method presented in the subsequent section or the method proposed in [197]. Both methods allow to avoid unnecessary distance computations. In the first case, the triangle inequality is exploited, and in the second one - a special data structure, the so-called kd-tree, is used.

Algorithm 3.2 k-means++ algorithm, [22]

```
1: Pick randomly the object $\mathbf{x} \in X$ and assume $\boldsymbol{\mu}_1 = \mathbf{x}$. Let $C = \{\boldsymbol{\mu}_1\}$.
2: for $j = 2$ to $k$ do
3:     For each object $\mathbf{x}$ compute the distance to the nearest centre from the set $C$,
       that is $u(\mathbf{x}) = \min_{\boldsymbol{\mu} \in C} \|\mathbf{x} - \boldsymbol{\mu}\|^2$.
4:     Substitute $\boldsymbol{\mu}_j$ with the object $\mathbf{x}$ picked randomly according to the distribution
       (3.5).
5:     Add $\boldsymbol{\mu}_j$ to the set of centres, $C \leftarrow C \cup \{\boldsymbol{\mu}_j\}$.
6: end for
7: Run the $k$-means algorithm.
```

Arthur and Vassilvitskii [22] demonstrate that under some conditions the algorithm converges in super-polynomial time. Beside this, they prove the following property.

Theorem 3.1.1 *[22] The expected value of the partition cost, computed according to equation (3.1), is delimited in case of k-means++ algorithm by the inequality*

$$\mathbb{E}(J) \leq 8(\ln k + 2)J_{opt} \tag{3.6}$$

where J_{opt} denotes the optimal value of the partition cost. □

The sequential nature is the main disadvantage of the algorithm. The proper initiation requires k-fold visiting of each element of the data set in order to choose the candidate cluster centres. The paper [30] contains a modification addressing this issue – the k-means|| algorithm , specially designed for very big data analysis. An additional advantage of the proposal is the possibility of paralleled implementation under the `MapReduce` paradigm[13].

3.1.4 Enhancing the efficiency of the k-means algorithm

Practical applications of k-means algorithm encounter the obstacle of the computational burden of multiple calculations of the distances $\|\mathbf{x}_i - \boldsymbol{\mu}_j\|$ for $i = 1, \ldots, m$, $j = 1, \ldots, k$. It turns out to be particularly expensive in case of large data sets where usually we have also a large number of clusters.

[13] A reader interested in this type of solutions is encouraged to consult the tutorial `https://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html`.

The simplest remedy, improving time complexity, is to compute the distance between a pair of objects by using the following identity

$$\|\mathbf{x}_i - \boldsymbol{\mu}_j\|^2 = \|\mathbf{x}\|^2 + \|\boldsymbol{\mu}_j\|^2 - 2\mathbf{x}_i^{\mathsf{T}}\boldsymbol{\mu}_j$$

The first and the second elements of this sum can be computed once for each $i = 1, \ldots, m$, $j = 1, \ldots, k$, while all the scalar products can be computed as XM^{T}, where M stands for the $k \times n$ matrix of centroids. The product of two matrices can be computed using `BLAS` package[14].

The classical approach to reduce the computational burden is to use special data structures, for example, k-dimensional binary search trees, so-called k-d trees [197], their variant called BBD-trees [23], metric trees [341], or the so-called R-trees [151]. An extensive review of methods of identification of the closest neighbour is contained in chapter [82], and a rich bibliography on the subject can be found on the Web page [233].

Another method consists in intelligent exploitation of the triangle inequality to eliminate superfluous computations of the distances [117]. One can achieve implementationally cheap algorithms with time complexity comparable to the algorithms of the former group.

The k-means algorithm may be parallelised quite easily. Assume that the data set has been split into P disjoint subsets $Y_p \subset X$, each of which is allocated to one of the processors. Assume further that global information is available to each of the processors on the identifiers of cluster centres and their coordinates and it has the form of a list of `<key, value>` pairs. Given this information, each processor performs locally two basic operations:

(a) It assigns elements of the set Y_p, allocated to it, to the clusters following the rule *winner takes all* (2.43), that is, it partitions Y_p into

$$Y_p = Y_{p,1} \cup \cdots \cup Y_{p,k}$$

and thereafter

(b) it computes the sums $s_{p,j} = \sum_{\mathbf{x} \in Y_{p,j}} \mathbf{x}$ and cardinalities of the "subclusters" $m_{p,j} = |Y_{p,j}|$, $j = 1, \ldots, k$, $p = 1, \ldots, P$.

The local lists of the form $<j, (s_{p,j}, m_{p,j})>$ are aggregated to the global list (by one of the processors), whereby the coordinates of the j-th centre are computed as follows:

$$\boldsymbol{\mu}_j = \frac{1}{m_j} \sum_{p=1}^{P} s_{p,j} \tag{3.7}$$

where $m_j = \sum_{p=1}^{P} m_{p,j}$. Both summations can be carried out in common **for** the loop, and thereafter the resulting sums can be divided by one another.

[14] Its source codes are available e.g. from `http://people.sc.fsu.edu/~jburkardt/c_src/blas3/blas3.html`.

The described steps are easy to implement within the `MapReduce` framework, see e.g. [389]. The step (a) fits the specification of the `map` function, the step (b) – that of the `combine` function, whereas the equation (3.7) may be embedded in the body of the `reduce` function. Under practical settings, the functions `map` and `combine` are integrated into one function. By extending this function according to the description provided in Section 3.1.4 one can significantly reduce the number of calls of the record-group distance by computing the function in step (a), thereby significantly reducing the execution time.

k-means algorithm possesses also hardware implementations, which are particularly useful for huge data analysis. A short review of such approaches is contained in [187].

An interesting attempt to improve the properties of the standard k-means algorithm was made with the development of `ISODATA` algorithm (*Iterative Self-Organizing Data Analysis Technique*). Its authors, Ball i Hall, enriched the base algorithm with the possibility of combining similar low cardinality clusters and with the possibility of splitting high cardinality clusters. Hence, `ISODATA`, features adaptive choice of the number of clusters. The details of the algorithm can be found in the monograph [337] and in [35].

Bradley, Benett and Bemiriz introduced in [57] the concept of a balanced k-means algorithm in which they require each cluster to contain at least $\widetilde{n}$ objects.

Wagstaff *et al.* proposed in [355] a semi-supervised variant of the classical k-means algorithm.

3.1.5 Variants of the *k*-means algorithm

3.1.5.1 *On line* variant of the *k*-means algorithm

For big data analysis, the so-called *on line* version of k-means algorithm has been developed. Its idea is outlined in the pseudo-code 3.3.

Algorithm 3.3 *On line* version of the k-means algorithm

```
1. Initialise (e.g. randomly) $k$ prototypes.
2. Select (according to the assumed probability distribution $p(\mathbf{x})$) an element $\mathbf{x} \in X$.
3. Determine the winner, that is, the prototype $\boldsymbol{\mu}_{s(\mathbf{x})}$ that is the closest to the considered object $\mathbf{x}$.
4. Modify the coordinates of the winning prototype, i.e.
       $\boldsymbol{\mu}_{s(\mathbf{x})} \leftarrow \boldsymbol{\mu}_{s(\mathbf{x})} + \alpha(\mathbf{x} - \boldsymbol{\mu}_{s(\mathbf{x})})$    (3.8)
   where $\alpha \in (0, 1]$ is called the learning coefficient
5. Go to step (2) if the termination condition is not met
```

The terminal condition of this algorithm is defined as stabilisation of prototypes, meaning that when each coordinate of the prototype before and after the modification does not differ by more than a pre-defined parameter ϵ.

The learning coefficient α can be either a constant or a function of (execution) time. It is known that in the first case the prototype coordinates $\boldsymbol{\mu}_j(t)$ at iteration t behave like the exponentially falling mean of the signals $\mathbf{x}^{(j)}(t)$, for which the prototype $\boldsymbol{\mu}_j(t)$ was a winner, that is

$$\boldsymbol{\mu}_j(t) = (1-\alpha)^t \boldsymbol{\mu}_j(0) + \alpha \sum_{i=1}^{t} (1-\alpha)^{t-i} \mathbf{x}^{(j)}(t)$$

The above equation implies that the current value of the prototype is most strongly influenced by the current observation, while the impact of the preceding observations decays exponentially with time. This property is the source of instability of the algorithm. Even after presenting a large number of observations, the next observation can radically impact the coordinates of the winning prototype.

MacQueen proposed in [240] to reduce the α coefficient hyperbolically with time, i.e.

$$\alpha(t) = \frac{\alpha_0}{t}, \qquad t \geq 1 \tag{3.9}$$

In this case, the vector $\boldsymbol{\mu}_j(t)$ is the arithmetic mean of the presented observations, that is:

$$\boldsymbol{\mu}_j(t) = \boldsymbol{\mu}_j(t-1) + \alpha(t)\Big(\mathbf{x}(t)^{(t)} - \boldsymbol{\mu}_j(t-1)\Big) = \frac{1}{t}(\mathbf{x}^{(j)}(1) + \cdots + \mathbf{x}^{(j)}(t))$$

In fact,

$$\boldsymbol{\mu}_j(1) = \boldsymbol{\mu}_j(0) + \alpha(1)(\mathbf{x}^{(j)}(1) - \boldsymbol{\mu}_j(0)) = \mathbf{x}^{(j)}(1)$$

$$\boldsymbol{\mu}_j(2) = \boldsymbol{\mu}_j(1) + \alpha(2)(\mathbf{x}^{(j)}(2) - \boldsymbol{\mu}_j(1)) = \tfrac{1}{2}(\mathbf{x}^{(j)}(1) + \mathbf{x}^{(j)}(2))$$

etc.

Another scenario of reducing the learning coefficient was suggested in [136]:

$$\alpha(t) = \alpha_p \left(\frac{\alpha_k}{\alpha_p}\right)^{\frac{t}{t_{max}}} \tag{3.10}$$

where α_p, α_k mean the initial and the final value of the coefficient with $alpha_p > \alpha_k$, and t_{max} is the maximal number of iterations. Further variants are discussed by Barbakh and Fyfe in [38].

3.1.5.2 Bisection variant of the *k*-means algorithm

Practical experience indicates that clusters obtained via agglomerative data analysis are generally of higher quality that those generated by the k-means algorithm. To increase the quality of the latter, Steinbach, Karypis and Kumar [325] proposed the so-called bisection variant of the k-means algorithm. Its idea (see pseudo-code 3.4) consists in initial division of the data set X into two clusters. Subsequently, one of the clusters is selected to be partitioned (again using k-means algorithm) into two new clusters. The process is repeated till a satisfactory number of clusters is obtained. Though diverse criteria may be applied to choose a cluster for partitioning, but, as the authors claim, [325], good results are obtained if one takes the largest cardinality cluster.

Algorithm 3.4 Bisectional k-means algorithm

```
1. Initialisation. Split the set $X$ into two clusters.
2. Choose a cluster $C$ for further partitioning (bisection).
3. Split the set $C$ into two subsets using $k$-means algorithm.
4. Repeat step (3) a fixed number of times (parameter ITER) till a partition of the
   highest quality (homogeneity) is obtained.
5. Repeat steps 2 – 3 till a proper number of clusters is obtained.
```

It is crucial to update cluster centres sequentially in the algorithm used in step (3), which leads to better results. Expressing it differently, if the cluster C_j of cardinality n_j and centre vector $\boldsymbol{\mu}_j$ is extended by the element $\mathbf{x}^*$, then the coordinates of the centre vector are updated according to the equation

$$\boldsymbol{\mu}_{jl} = \frac{n_j \cdot \boldsymbol{\mu}_{jl} + \mathbf{x}^*_l}{n_j + 1}, \quad l = 1, \dots, n$$

Though k-means algorithm guarantees reaching the local minimum of the function (3.1), it is not more the case in the incremental version where it is applied "locally" to a selected cluster. Nonetheless the bisectional variant produces clusters of similar size and of high quality. Here, the quality is measured as the averaged entropy

$$E = \frac{1}{m} \sum_{j=1}^{k} n_j E_j$$

where n_j is the cardinality of the j-th cluster, E_j is the entropy of this cluster

$$E_j = -\sum_{t=1}^{k} p_{tj} \log p_{tj}$$

and p_{tj} denotes the probability that an element of, j-th cluster will be classified into the cluster C_t.

3.1.5.3 Spherical *k*-means algorithm

This is a variant of the k-means algorithm, developed for clustering of text documents, [102], [99], [98]. It is assumed that documents are represented by vectors with components reflecting the frequencies of word occurrences in respective documents. These vectors are deemed to be normalized, that is $\|\mathbf{x}\| = 1, \mathbf{x} \in X$, hence they belong to the surface of a unit sphere – which implies the name of the algorithm. In such a case, the squared distance between the i-th document and the j-th prototype is equal

$$d^2(\mathbf{x}_i, \boldsymbol{\mu}_j) = \|\mathbf{x}_i - \boldsymbol{\mu}_j\|^2 = \|\mathbf{x}\|^2 - 2\mathbf{x}_i^{\mathsf{T}}\boldsymbol{\mu}_j + \|\boldsymbol{\mu}_j\|^2 = 2(1 - \mathbf{x}_i^{\mathsf{T}}\boldsymbol{\mu}_j) \tag{3.11}$$

where the product $\mathbf{x}_i^{\mathsf{T}}\boldsymbol{\mu}_j$ equals to the cosine of the angle between the (normalised) vectors representing both objects. Hence, the minimisation of the error (3.1) corresponds to maximisation of the expression

$$J_s(U, M) = \sum_{i=1}^{m}\sum_{j=1}^{k} u_{ij}\mathbf{x}_i^{\mathsf{T}}\boldsymbol{\mu}_j \tag{3.12}$$

The search for the optimal partition of the set of documents could follow the k-means algorithm, i.e. the prototype $\boldsymbol{\mu}_j$ components could be determined according to equation (3.3), and the assignment of objects to clusters could be governed by the "winner-takes-all" rule. However, the vector $\boldsymbol{\mu}_j$, determined in this way, would not be a unit vector, as required. Therefore, it is replaced with the direction vector $\mathbf{c}_j$, as defined by equation (3.13):

$$\mathbf{c}_j = \frac{\boldsymbol{\mu}_j}{\|\boldsymbol{\mu}_j\|} = \frac{\sum_{\mathbf{x}_i \in C_j} \mathbf{x}_i}{\|\sum_{\mathbf{x}_i \in C_j} \mathbf{x}_i\|} \tag{3.13}$$

The Cauchy -Schwartz -Buniakowski inequality implies for each unit vector $\mathbf{z}$:

$$\sum_{\mathbf{x} \in C_j} \mathbf{x}^{\mathsf{T}}\mathbf{z} \leq \sum_{\mathbf{x} \in C_j} \mathbf{x}^{\mathsf{T}}\mathbf{c}_j \tag{3.14}$$

so that $\mathbf{c}_j$ is the most similar vector (with respect to the cosine similarity measure) to the vectors forming the cluster C_j. This means that $\mathbf{c}_j$ represents the leading *topic* in the cluster C_j.

Knowing the main (leading) topic of cluster C_j, one can define the *consistency* of this cluster, that is, its quality, measured as the sum of similarities of the collected documents on this topic:

$$q(C_j) = \begin{cases} \sum_{\mathbf{x}_i \in C_j} \mathbf{x}_i^{\mathsf{T}}\mathbf{c}_j & \text{if } C_j \neq \emptyset \\ 0 & \text{otherwise} \end{cases} \tag{3.15}$$

The property $\boldsymbol{\mu}_j = \frac{1}{n_j}\sum_{\mathbf{x} \in C_j} \mathbf{x}^{\mathsf{T}}$, combined with the equation (3.13), implies:

$$q(C_j) = \sum_{\mathbf{x}_i \in C_j} \mathbf{x}_i^{\mathsf{T}}\mathbf{c}_j = n_j\boldsymbol{\mu}^{\mathsf{T}}c_j = n_j\|\boldsymbol{\mu}_j\|\mathbf{c}_j^{\mathsf{T}}\mathbf{c}_j = n_j\boldsymbol{\mu}_j = \|\sum_{\mathbf{x} \in C_j} \mathbf{x}\| \tag{3.16}$$

which points at an additional interpretation of the cluster quality index, [99], [98].

The quality of the partitioning $C = \{C_1, \ldots, C_k\}$ is now defined by the formula:

$$Q(\{C_1, \ldots, C_k\}) = \sum_{j=1}^{k} q(C_j) = \sum_{j=1}^{k} \sum_{\mathbf{x}_i \in C_j} \mathbf{x}_i^{\mathrm{T}} \mathbf{c}_j \tag{3.17}$$

constituting an analogue of quality index (3.1), but with the amendment that we seek now such a partition $\{C_1, \ldots, C_k\}$ that maximises the indicator (3.17). An efficient heuristic has been proposed to solve this problem. It is presented as the pseudo-code 3.5.

Algorithm 3.5 Spherical k-means algorithm

```
1: *Initialisation.* Create any partition of the set of documents $C^0 = \{C_1^0, \ldots, C_k^0\}$ and
   determine the topics $\mathbf{c}_j^0, j = 1, \ldots, k$. Set the iteration counter $t = 0$.
2: For each document $\mathbf{x}_i$ find the closest topic (with respect to the cosine measure).
   Create the new partition
       $C_j^{t+1} = \{\mathbf{x} \in X : \mathbf{x}^{\mathrm{T}}\mathbf{c}_j^t \geq \mathbf{x}^{\mathrm{T}}\mathbf{c}_l^t, l = 1, \ldots, m\}, j = 1, \ldots, k$
   i.e. $C_j^{t+1}$ contains the documents that are most similar to the topic $\mathbf{c}_j$. If a document
   happens to be equally similar to several topics, assign it to any of them randomly.
3: Update the characteristics of the topics $\mathbf{c}_j^{t+1}$ as indicated by the equation (3.13),
   substituting $C_j$ with $C_j^{t+1}$.
4: If the stop condition does not hold, set the counter $t$ to $t+1$ and return to step 2.
```

It has been demonstrated in [102] that for each $t \geq 0$, the algorithm 3.5 preserves the following property

$$Q\big(\{C_j^{t+1}\}_{j=1}^k\big) \geq Q\big(\{C_j^t\}_{j=1}^k\big) \tag{3.18}$$

The paper [97] presents several algorithms for improving the performance for big document collections.

The spherical algorithm exhibits the properties similar to those of the previously discussed k-means algorithm. In particular, it is sensitive to initialisation and the optimisation process can get stuck at a local optimum.

The significant difference between the spherical and non-spherical k-means algorithms is related to the shape of the clusters [102]. In case of spherical algorithm, the border between the clusters C_j and C_l, represented by topics $\mathbf{c}_j$, $\mathbf{c}_l$, is described by the equation

$$\mathbf{x}^{\mathrm{T}}(\mathbf{c}_j - \mathbf{c}_l) = 0$$

This is an equation of a hyperplane, passing through the origin of the coordinate system. Such a hyperplane cuts the unit sphere at the equator. In case of standard k-means algorithm, the distance between i-th point and j-th cluster is equal

$$\|\mathbf{x}_i - \boldsymbol{\mu}_j\|^2 = (\mathbf{x}_i - \boldsymbol{\mu}_j)^{\mathrm{T}}(\mathbf{x}_i - \boldsymbol{\mu}_j) = \mathbf{x}_i^{\mathrm{T}}\mathbf{x}_i - 2\mathbf{x}_i^{\mathrm{T}}\boldsymbol{\mu}_j + \boldsymbol{\mu}_j^{\mathrm{T}}\boldsymbol{\mu}_j = \mathbf{x}_i^{\mathrm{T}}\mathbf{x}_i - 2(\mathbf{x}_i^{\mathrm{T}}\boldsymbol{\mu}_j - \frac{1}{2}\boldsymbol{\mu}_j^{\mathrm{T}}\boldsymbol{\mu}_j)$$

Hence, we have the final form of the border equation:

$$\|\mathbf{x}_i - \boldsymbol{\mu}_j\|^2 - \|\mathbf{x}_i - \boldsymbol{\mu}_l\|^2 = 0 \equiv \mathbf{x}^{\mathtt{T}}(\boldsymbol{\mu}_j - \boldsymbol{\mu}_l) = \frac{1}{2}(\boldsymbol{\mu}_j^{\mathtt{T}}\boldsymbol{\mu}_j - \boldsymbol{\mu}_l^{\mathtt{T}}\boldsymbol{\mu}_l)$$

It is also an equation of a hyperplane, but it intersects the sphere at any place.

More remarks about the algorithm and its various implementations (batch and incremental ones) can be found in the fourth chapter of the book [209]. The paper [390] presents an adaptation of the algorithm for the processing of large data collections.

3.1.5.4 KHM: the harmonic k-means algorithm

Let us rewrite the equation (3.1) in the equivalent form

$$J(U, M) = \sum_{i=1}^{m} \min_{j=1,\dots,k}, \|\mathbf{x}_i - \boldsymbol{\mu}_j\|^2 \tag{3.19}$$

Zhang [383] found that the properties of the function $\min(a_1, \dots, a_k)$, where $a_j, j = 1, \dots, k$ are positive real numbers can be quite faithfully represented by the harmonic mean of these numbers

$$m_h(a_1, \dots, a_k) = \frac{k}{\sum_{j=1}^{k} \frac{1}{a_j}}$$

Contour diagrams of both functions are presented in Figure 3.4.

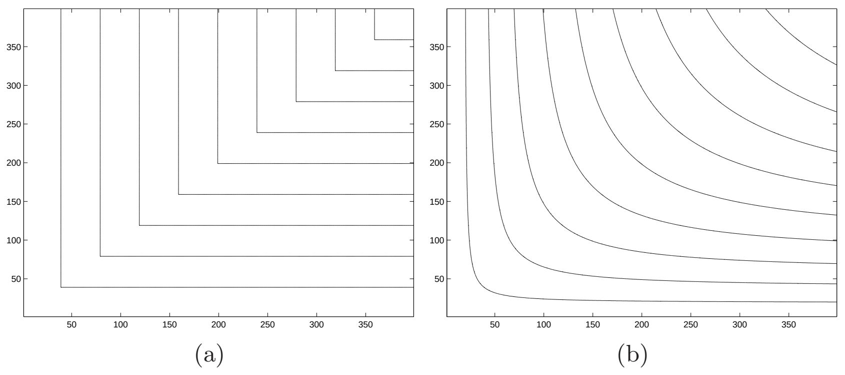


Fig. 3.4. Contour diagrams of the function: (a) $\min(a_1, a_2)$, and (b) $m_h(a_1, a_2)$. It was assumed in both cases that $a_1, a_2 \in (0, 400]$. Horizontal axis: a_1 value, vertical axis: a_2 value, the contour lines were drawn for equidistant values.

Similarities between both functions can be strengthened or weakened by introduction of an additional parameter – the power exponent, to which the arguments of harmonic mean are raised. Finally, by replacing in equation (3.19) the minimum operator by the harmonic mean operator, we obtain a new criterion function of the form

$$J_h(U,M) = \sum_{i=1}^{m} \frac{k}{\sum_{j=1}^{k} \|\mathbf{x}_i, \boldsymbol{\mu}_j\|^{-p}} \tag{3.20}$$

where $p \geq 2$ is a parameter. It is suggested to set $p = 3.5$, [383].

So far, we have been dealing with a sharp (crisp) partition. But the function (3.20) induces a fuzzy partition, in which the membership of i-th object in j-th cluster is a number $u_{ij} \in [0,1]$. It is determined from the equation, [383]

$$u_{ij} = \frac{\|\mathbf{x}_i - \boldsymbol{\mu}_j\|^{-2-p}}{\sum_{j=1}^{k} \|\mathbf{x}_i - \boldsymbol{\mu}_j\|^{-2-p}} \tag{3.21}$$

To compute the coordinates of the cluster centres, the weight $w(\mathbf{x}_i)$, occurring in the third step of the algorithm 2.2, presented on page 48, needs to be determined first.[15]:

$$w(\mathbf{x}_i) = \frac{\sum_{j=1}^{k} \|\mathbf{x}_i - \boldsymbol{\mu}_j|^{-2-p}}{\left(\sum_{j=1}^{k} \|\mathbf{x}_i - \boldsymbol{\mu}_j\|^{-p}\right)^2} \tag{3.22}$$

This weight promotes the points that are distant from all the centres, which enforces relocation of these centres in such a way as to cover all data.

If $\mathbf{x}_i \approx \boldsymbol{\mu}_j$, then the original distance is replaced by the value $\max\{\|\mathbf{x}_i - \boldsymbol{\mu}_j\|, \epsilon\}$, where ϵ is a "sufficiently small number", as suggested by [383].

Fuzzy cluster membership of objects and weights of these objects allow for the computation of the centre coordinates – see the third step of the algorithm 2.2

$$\boldsymbol{\mu}_j = \frac{\sum_{i=1}^{m} u_{ij} w(\mathbf{x}_i)\mathbf{x}_i}{\sum_{i=1}^{m} u_{ij} w(\mathbf{x}_i)} \tag{3.23}$$

This algorithm is less sensitive to initialisation [383]. However, it can still converge to a local minimum.

Hamerly and Elkan proposed in [160] two modifications, called Hybrid-1 i Hybrid-2. The first hybrid is a combination of the k-means algorithm with `KHM`. The split into clusters is crisp (an object belongs to the cluster with the closest centre), but each object is assigned a weight according to the formula (3.22). The membership degree of an object to a cluster in the second hybrid is determined by the equation (3.21), but the weights are constant, $w(\mathbf{x}_i) = 1, i = 1, \ldots, m$. None of these hybrids considered has a quality superior to `KHM` algorithm. However, their analysis indicates that the most important factor is the fuzzy membership u_{ij}. Hence, Hybrid-2 proved to be only a little bit inferior to the original KHM algorithm. The introduction of weights, on the other hand, improves the properties of the k-means algorithm.

[15] The plain k-means algorithm assumes $w(\mathbf{x}_i) = 1$

Experiments presented in papers [160], [383], show that KHM provides with higher quality results than not only the k-means algorithm but also the fuzzy k-means algorithm, presented in Section 3.3. Further improvements of the algorithm, based on meta-heuristics, are presented in [375] and in the literature cited therein.

3.1.5.5 Kernel based *k*-means algorithm

The idea of the algorithm is to switch to a multidimensional feature space $\mathcal{F}$ and to search therein for prototypes $\boldsymbol{\mu}_j^{\Phi}$ minimizing the error

$$J_2^{\Phi}(U) = \sum_{i=1}^{m} \min_{1 \le j \le k} \|\Phi(\mathbf{x}_i) - \boldsymbol{\mu}_j^{\Phi}\|^2 \tag{3.24}$$

where $\Phi\colon \mathbb{R}^n \to \mathcal{F}$ is a non-linear mapping of the space X into the feature space. This is the second variant of application of the kernel functions in cluster analysis, according to classification from the Section 2.5.4. In many cases, switching to the space $\mathcal{F}$ allows to reveal clusters that are not linearly separable in the original space.

In analogy to the classical k-means algorithm, the prototype vectors are updated according to the equation

$$\boldsymbol{\mu}_j^{\Phi} = \frac{1}{n_j} \sum_{\mathbf{x}_i \in C_j} \Phi(\mathbf{x}_i) = \frac{1}{n_j} \sum_{i=1}^{m} u_{ij} \Phi(\mathbf{x}_i) \tag{3.25}$$

where $n_j = \sum_{i=1}^m u_{ij}$ is the cardinality of the j-th cluster, and u_{ij} is the function, allocating objects to clusters, i.e. $u_{ij} = 1$ when $\mathbf{x}_i$ is an element of the j-th cluster and $u_{ij} = 0$ otherwise. A direct application of this equation is not possible, because the function Φ is not known. In spite of this, it is possible to compute the distances between the object images and prototypes in the feature space, making use of equation (2.48). The reasoning runs as follows:

$$\begin{aligned}
\|\Phi(\mathbf{x}_i) - \boldsymbol{\mu}_j^{\Phi}\|^2 &= \big(\Phi(\mathbf{x}_i) - \boldsymbol{\mu}_j^{\Phi}\big)^{\mathsf{T}} \big(\Phi(\mathbf{x}_i) - \boldsymbol{\mu}_j^{\Phi}\big) \\
&= \Phi(\mathbf{x}_i)^{\mathsf{T}} \Phi(\mathbf{x}_i) - 2\Phi(\mathbf{x}_i)^{\mathsf{T}} \boldsymbol{\mu}_j^{\Phi} + (\boldsymbol{\mu}_j^{\Phi})^{\mathsf{T}} \boldsymbol{\mu}_j^{\Phi} \\
&= \Phi(\mathbf{x}_i)^{\mathsf{T}} \Phi(\mathbf{x}_i) - \frac{2}{n_j} \sum_{h=1}^{m} u_{hj} \Phi(\mathbf{x}_i)^{\mathsf{T}} \Phi(\mathbf{x}_h) + \\
&\quad + \frac{1}{n_j^2} \sum_{r=1}^{m} \sum_{s=1}^{m} u_{rj} u_{sj} \Phi(\mathbf{x}_r)^{\mathsf{T}} \Phi(\mathbf{x}_s) \\
&= k_{ii} - \frac{2}{n_j} \sum_{h=1}^{m} u_{hj} k_{hi} + \frac{1}{n_j^2} \sum_{r=1}^{m} \sum_{s=1}^{m} u_{rj} u_{sj} k_{rs}
\end{aligned} \tag{3.26}$$

where, like in Section 2.5.4, $k_{ij} = \Phi(\mathbf{x}_i)^{\mathsf{T}}\Phi(\mathbf{x}_j) = \mathsf{K}(\mathbf{x}_i, \mathbf{x}_j)$. If we denote with $\mathbf{u}_j$ the j-th column of the matrix U and substitute $\widetilde{\mathbf{u}}_j = \mathbf{u}_j/\|\mathbf{u}_j\|_1$, then the above equation can be rewritten in a closed form:

$$\|\Phi(\mathbf{x}_i) - \boldsymbol{\mu}_j\|^2 = k_{ii} - 2(\widetilde{\mathbf{u}}_j^{\mathsf{T}} K)_i + \widetilde{\mathbf{u}}_j^{\mathsf{T}} K \widetilde{\mathbf{u}}_j \tag{3.27}$$

where K is a Gram matrix with elements k_{ij}, and the scalar $(\widetilde{\mathbf{u}}_j^{\mathsf{T}} K)_i$ denotes the i-th component of the vector $(\widetilde{\mathbf{u}}_j^{\mathsf{T}} K)$.

In this way, one can update the elements of the matrix U without determining the prototypes explicitly. This is possible, because $\mathbf{x}_i$ is assigned to the cluster minimising the above-mentioned distance, i.e.

$$u_{ij} = \begin{cases} 1 \text{ if } \|\Phi(\mathbf{x}_i) - \boldsymbol{\mu}_j^{\Phi}\|^2 = \min_{1 \le t \le k} \|\Phi(\mathbf{x}_i) - \boldsymbol{\mu}_t^{\Phi}\|^2 \\ 0 \text{ otherwise} \end{cases} \tag{3.28}$$

Though it is not possible to use the equation (3.25) directly, one can determine (in the original feature space X) the approximate cluster prototypes by assuming $\boldsymbol{\mu}_j$ to be the object $\mathbf{x}^j$ matching the condition [386]

$$\mathbf{x}_i^j = \arg\min_{\mathbf{x}_i \in C_j} \|\Phi(\mathbf{x}_i) - \boldsymbol{\mu}_j^{\Phi}\|^2 \tag{3.29}$$

Prototypes, defined in this way, are in fact medoids (see the next section). Another method of determining the prototypes is presented in Section 3.3.5.6.2.

The algorithm is summarized in the form of the pseudo-code 3.6. It needs to be stressed that, like the k-means algorithm, its kernel based variant is also sensitive to initialisation, meaning the 2^{nd} step of the algorithm. The simplest way to initialise it is to assign $k-1$ (randomly selected) objects to $k-1$ different clusters and to allocate the remaining $m-k+1$ objects to the k-th cluster. Another method is to assume the partition returned by the classical k-means algorithm. It guarantees that at least a part of the objects will belong to proper clusters and the algorithm 3.6 will only modify erroneous assignments.

Algorithm 3.6 Kernel-based k-means algorithm

```
1: Choose a kernel function $\mathsf{K}$ and compute the elements of the Gram matrix $k_{ij} = \mathsf{K}(\mathbf{x}_i, \mathbf{x}_j)$ for the set of objects $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$.
2: Compute the initial allocation of elements among clusters.
3: For each object $\mathbf{x}_i \in X$ compute its distance from each cluster by applying equation (3.26), and assign $\mathbf{x}_i$ to the closest cluster
4: Repeat step (3) till none of $u_{ij}$ values changes.
5: Determine the approximate prototypes according to equation (3.29).
```

The presented algorithm clusters correctly the data similar to the ones in Figure 3.5, and, as reported by authors of [203], even in case of such sets as `iris` one gets better clustering performance (compared to k-means algorithm). An *on line* version of this algorithm was presented by Schölkopf, Smola and Müller in [306], and its modifications are presented in [138].

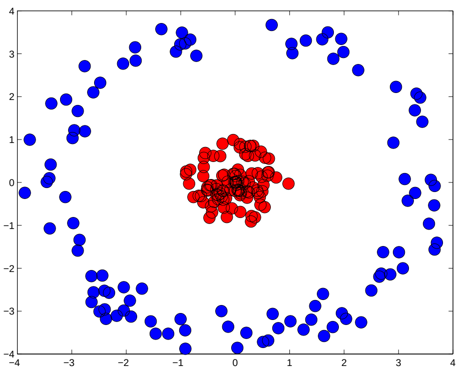


Fig. 3.5. Results of application of the kernel-based variant of the k-means algorithm. Object-to-cluster allocation is indicated by different colours. $\sigma = 1$ was assumed.

3.1.5.6 *k*-medoids algorithm

The k-means algorithm is formally applicable only if the dissimilarity between pairs of objects is equal to the square of their Euclidean distance. This means that the features describing the object properties must be measured on quantitative scale, e.g. on the ratio scale. In such a case, the objects can be represented as n-dimensional vectors with real-valued components. Beside this, the partition cost,given by equation (3.1) depends on the maximal distances between the prototype $\boldsymbol{\mu}_i$ and the object of the cluster represented by this prototype. Hence, the k-means algorithm is not resistant to the presence of outliers. Furthermore, the centres $\boldsymbol{\mu}_i$ are abstract objects not belonging to the set $\mathfrak{X}$. To avoid these disadvantages, one may measure the partition cost as follows:

$$J_{med}(\mathfrak{p}_1, \ldots, \mathfrak{p}_k) = \min_{1 \leq j \leq k} \sum_{i \in C_j} d(\mathfrak{x}_i, \mathfrak{p}_j) \tag{3.30}$$

where $\mathfrak{p}_1, \ldots, \mathfrak{p}_k \in \mathfrak{X}$ are cluster centres, called also prototypes, examples, exemplars, medoids or just centres, and d: $\mathfrak{X} \times \mathfrak{X} \to \mathbb{R}$ is a dissimilarity measure. There is no need for this measure to be symmetric or even metric (as required for Euclidean distance or its generalisations).

The k-medoids (or k-centres) algorithm aims at such a choice of centres $\{\mathfrak{p}_1, \ldots, \mathfrak{p}_k\} \subset \mathfrak{X}$ for which the index J_{med} reaches its minimum. By introducing the dissimilarity measure $d_{ij} = d(\mathfrak{x}_i, \mathfrak{p}_j)$ we broaden the applicability of the algorithm requiring only that for each pair of objects from the set $\mathfrak{X}$ it is possible to compute their dissimilarity. Working with dissimilarity matrix has an additional advantage: its dimension depends on the number of objects only.

As the object representation stops playing any role here, the practical implementations of the algorithm makes use only of the vector $\mathbf{c}$ having the elements c_i, indicating to which example (and in this way to which cluster) the i-th object belongs. More precisely:

$$c_i = \begin{cases} j \text{ if } \mathfrak{x}_i \in C_j \\ i \text{ if } \mathfrak{x}_i \text{ is an exemplar} \end{cases} \tag{3.31}$$

An elementary implementation of the k-medoids algorithm is demonstrated by the pseudo-code 3.7, see e.g. [164, sec. 14.3.10].

Algorithm 3.7 k-medoids algorithm

```
Require: Dissimilarity matrix $D = [d_{ij}]_{m\times m}$, number of clusters $k$
1: Select the subset $\mathcal{K} \subset \{1,\dots,m\}$. Its elements are pointers to examples (prototypes)
2: while (not termination condition) do
3:     Assign objects to clusters using the rule
           $$c_i = \begin{cases} \arg\min_{j\in\mathcal{K}} d_{ij} & \text{if } i \notin \mathcal{K} \\ i & \text{otherwise} \end{cases}, \quad i = 1,\dots,m \tag{3.32}$$
4:     Update the examples, that is
           $$j_r^* = \arg\min_{t:\, c_t = r} \sum_{t':\, c_{t'} = r} d_{tt'}, \quad r = 1,\dots,k \tag{3.33}$$
5: end while
6: If the index value remained unchanged after testing $m$ objects – Stop. Otherwise return to step 2.
```

In the simplest case the examples are picked at random, but very good results can be achieved by adapting the methods from Section 3.1.3. In particular, one can start with selection of k most dissimilar objects.

The equation (3.32) tells us that the object i is assigned to the least dissimilar example from the set $\mathcal{K}$. On the other hand, the equation (3.33) states that for a set of objects sharing a common example we select as the new example such an object for which the sum of dissimilarities to other objects of the cluster is the lowest. Like the k-means algorithm, the k-medoids algorithm stops when the new examples are identical with those of the previous iteration.

In spite of its superficial simplicity, the algorithm is much more time-consuming than k-means algorithm: determining a new example requires $O(|C_r|)$ operations, and the cluster allocation updates – $O(mk)$ comparisons.

Kaufman and Rousseeuw initiated research on elaboration of efficient method of determination of examples. In [201] they proposed two algorithms: PAM (*Partitioning Around Medoids*) and CLARA (*Clustering LARge Applications*). PAM follows the just described principle, that is- it seeks in $\mathfrak{X}$ such *medoid* objects which minimise the index (3.1). This approach is quite expensive. The big data analysis algorithm CLARA uses several (usually five) samples consisting of $40 + 2k$ objects and applies the PAM algorithm to each of the samples to obtain a set of proposed sets of medoids. Each proposal is evaluated using the index (3.1); the set yielding the lowest value of the criterion function is chosen.

The next improvement was the CLARANS (*Clustering Large Applications based upon RANdomized Search*), [270] – an algorithm with quadratic complexity in the number of objects m. The algorithm construes a graph of sets of k *medoids*; two graph nodes are connected by an edge if the assigned sets differ by exactly one element. The neighbours are generated using random search techniques. Another variant of the k-centres algorithm was proposed in [278].

In Figure 3.6 allocations of objects to examples for the data sets `data3_2` and `data6_2` are presented.

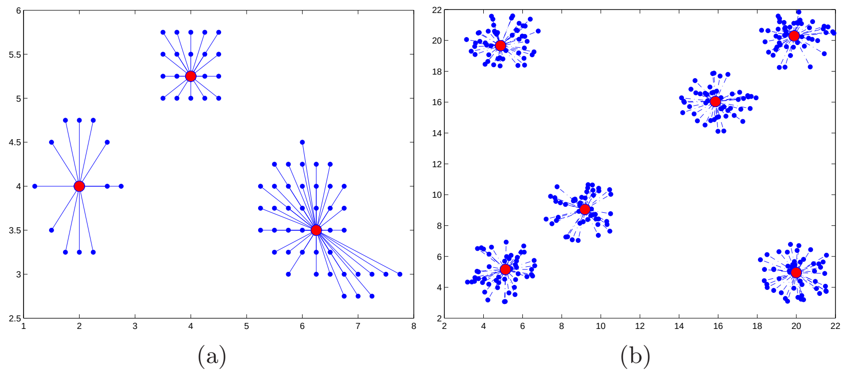


Fig. 3.6. Clustering of separable data sets with the k-medoids algorithm: (a) the set `data3_2`, (b) the set `data6_2`. Cluster examples are marked with red dots.

Remark 3.1.1 *The k-medoids algorithm should not be confused with the k-medians algorithm, proposed to make the k-means algorithm resistant against outliers. In the k-medians algorithm the Euclidean distance is replaced with the Manhattan distance, assuming $p = 1$ in the equation (3.2), [57], [191]. In such a case the cluster centres are determined from the equation*

$$\sum_{\mathbf{x}_i \in C_j} \frac{\partial}{\partial \mu_{jl}} |x_{il} - \mu_{jl}| = 0 \Rightarrow \sum_{\mathbf{x}_i \in C_j} sgn(x_{il} - \mu_{jl}) = 0$$

i.e. $\mu_j l$ are the medians of the respective components of the vectors $\mathbf{x}_i$ assigned to the cluster C_j.

This algorithm plays an important role in operations research, in particular in the choice of location of service centres[16]. □

3.1.5.7 *k*-modes algorithm

Huang presented in [183] an adaptation of the k-means algorithm for cases where the features describing the objects are measured on nominal scale. Examples of

[16] See e.g. N. Mladenović, J. Brimberg, P. Hansen, J.A. Moreno-Pérez. The p-median problem: A survey of meta-heuristic approaches. *European J. of Op. Res.*, 179(3), 2007, 927–939.

such features are: sex, eye colour, hair colour, nationality etc. Each feature takes usually a value from a small value set; for example the value of the feature "eye colour" can be: "blue", "brown", "black" etc. By replacing these values with consecutive integers we can assign each real object $\mathfrak{x}_i$ a concise representation in terms of a vector $\mathbf{x}_i$, elements of which x_{il} are integers *pointing at* the proper value of the i-th feature. Now the dissimilarity of the pair of objects i and j is defined as (see Section 2.2.2):

$$d_{ij} = d(\mathbf{x}_i, \mathbf{x}_j) = \sum_{l=1}^{n} \delta(x_{il}, x_{jl}) \tag{3.34}$$

where

$$\delta(x_{il}, x_{jl}) = \begin{cases} 1 \text{ if } x_{il} = x_{jl} \\ 0 \text{ otherwise} \end{cases}$$

hence $d(\mathbf{x}_i, \mathbf{x}_j)$ counts the number of differences between the compared objects.

The mode of the set C is defined as an object $\mathbf{m}$ (not necessarily from the set $\mathfrak{X}$) minimizing the rating:

$$d(C, \mathbf{m}) = \sum_{\mathbf{x} \in C} d(\mathbf{x}, \mathbf{m}) \tag{3.35}$$

Huang noted in [183] the following property that allows to identify the mode of the set C quickly:

Lemma 3.1.1 *Let $Dom(A_l)$ denote the set of values of the feature A_l and let c_{ls} denote the number of objects in the set C in which the feature l takes on the value s. Vector $\mathbf{m}$ with components matching the condition*

$$m_l = \underset{s \in Dom(A_l)}{\arg\max}\ c_{ls},\ l = 1, \ldots, n \tag{3.36}$$

is the mode of the set C. □

Let $\mathbf{m}_j$ denote the mode of the j-th cluster. The problem of k-modes consists in finding such modes $\mathbf{m}_1, \ldots, \mathbf{m}_k$ that the index

$$J_{mod}(\mathbf{m}_1, \ldots, \mathbf{m}_k) = \sum_{j=1}^{k} d(C_j, \mathbf{m}_j) = \sum_{j=1}^{k} \sum_{\mathbf{x}_i \in C_i} d(\mathbf{x}_i, \mathbf{m}_j) \tag{3.37}$$

is minimised. A method to solve this problem is presented in the form of the pseudo-code 3.8.

The subsequent theorem points at the difference between the k-medoids and the k-modes algorithms

Theorem 3.1.2 *Let $\mathfrak{p}_1, \ldots, \mathfrak{p}_k$ be a set of medoids and let $\mathbf{m}_1, \ldots, \mathbf{m}_k$ be the set of modes. Let both be computed using respective algorithms. Then*

$$J_{med}(\mathfrak{p}_1, \ldots, \mathfrak{p}_k) \leq 2J_{mod}(\mathbf{m}_1, \ldots, \mathbf{m}_k) \tag{3.38}$$

□

Algorithm 3.8 k-modes algorithm, [183]

```
Require: Matrix $X$ representing the set of objects, number of clusters $k$.
1: Determine the dissimilarities $d_{ij}$ for each pair of objects.
2: determine in any way $k$ modes.
3: while (not termination condition) do
4:    Using the dissimilarity measure 3.34 assign objects to the closest clusters
5:    Update mode of each cluster applying the Lemma 3.1.1.
6: end while
7: Return a partition into $k$ clusters.
```

Let us note however that the execution of the k-medoids algorithm requires only the dissimilarity matrix $D = [d_{ij}]$ while the k-modes algorithm insists on access to the matrix X with rows of which containing the characteristics of individual objects.

Huang considers in [183] the general case when the features describing objects are measured both on ratio scale and on nominal scale. Let us impose such an order on the features that the first n_1 features are measured on ratio scale while the remaining ones – on nominal scale. In such a case the cost of partition represented by the matrix $U = [u_{ij}]$ can be determined as follows:

$$J_p(U, M) = \sum_{j=1}^{k} \sum_{i=1}^{m} \Big[u_{ij} \sum_{l=1}^{n_1} (x_{il} - m_{jl})^2 + \gamma u_{ij} \sum_{l=n_1+1}^{n} \delta(x_{il}, m_{jl}) \Big] \tag{3.39}$$

where $\gamma > 0$ is a coefficient balancing both types of dissimilarities.

By substituting

$$P_j^r = \sum_{i=1}^{m} u_{ij} \sum_{l=1}^{n_1} (x_{il} - m_{jl})^2, \quad P_j^n = \gamma \sum_{i=1}^{m} u_{ij} \sum_{l=n_1+1}^{n} \delta(x_{il}, m_{jl})$$

we rewrite the equation (3.39) in the form

$$J_p(U, M) = \sum_{j=1}^{k} (P_j^r + P_j^n) \tag{3.40}$$

Optimisation of this index is performed iteratively, namely, starting with the initial centroid matrix M objects are assigned to clusters with the least differing centroids. The dissimilarity measure of the object $\mathbf{x}_i$ with respect to centroid $\mathbf{m}_i$ equals to

$$d(\mathbf{x}_i, \mathbf{m}_i) = \sum_{l=1}^{n_1} (x_{il} - m_{jl})^2 + \gamma \sum_{l=N_1+1}^{n} \delta(x_{ij}, m_{jl}) \tag{3.41}$$

Subsequently new cluster centres are determined. Their components being real numbers are computed as in the classic k-means algorithm, see equation (3.3), while for the nominal valued components the Lemma 3.1.1 is applied.

Huang commented in [183] on the difference between this algorithm (called by the author *k-prototypes*) and the algorithm CLARA, mentioned in the previous section :

> The major differences between CLARA and the k-prototypes algorithm are as follows: (1) CLARA clusters a large data set based on samples, whereas k-prototypes directly works on the whole data set. (2) CLARA optimises its clustering result at the sample level. A good clustering based on samples will not necessarily represent a good clustering of the whole data set if the sample is biased. The k-prototypes algorithm optimises the cost function on the whole data set. It guarantees at least a locally optimal clustering. (3) The efficiency of CLARA depends on the sample size. The larger and more complex the whole data set is, the larger the sample is required. CLARA will no longer be efficient when the sample size exceeds a certain range, say thousands of objects. The k-prototypes algorithm has no such limitations.

It is worth mentioning that the dissimilarity measure used here fulfils the triangle inequality, which implies that accelerations mentioned in Section 3.1.4 are applicable here.

3.2 EM algorithm

Instead of concentrating on a given set of objects, a clustering process may target a population from which the actually obtained set has been sampled. In this case one assumes that the population can be described as a sum of statistical models, where each of the models describes a cluster of objects. A statistical model may be a probability density function of occurrence of objects of the given cluster in the feature space.

Frequently, as a matter of convenience, it is assumed that the probability density distribution function that describes the set of observations X is a mixture of Gaussian distributions,

$$p(\mathbf{x}) = \sum_{j=1}^{k} p(C_j) p(\mathbf{x}|C_j; \boldsymbol{\mu}_j, \Sigma_j) \tag{3.42}$$

where $p(C_j)$ denotes the *a priori* probability of the object to belong to the j-th cluster, while $p(\mathbf{x}|C_j; \boldsymbol{\mu}_j, \Sigma_j)$ is the probability density of the n-dimensional Gaussian distribution with the vector of expected values $\boldsymbol{\mu}_j$ and covariance matrix Σ_j according to which the observations belonging to the j-th cluster are generated. Let us recall the formula for the probability density function of multidimensional (multivariate) Gaussian distribution:

$$p(\mathbf{x}|C_j; \boldsymbol{\mu}_j, \Sigma_j) = \frac{1}{(2\pi)^{n/n}\sqrt{|\Sigma_j|}} \exp\Big[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_j)^{\mathrm{T}} \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu}_j) \Big] \tag{3.43}$$

The assumption of Gaussian (or normal) distributions of elements in a cluster is a "classical" variant of the discussed approach but it is nonetheless sufficiently general. It is possible to demonstrate that every continuous and constrained probability density distribution can be approximated with any required precision by a mixture of Gaussian distributions [213]. Sample sets that were generated from a mixture of two normal distributions were presented already in Figure 2.13 on page 65. But, in general, mixtures of any type of distributions can be considered. A rich bibliography on this subject can be found on the Web page of David Dowe `http://www.csse.monash.edu.au/~dld/mixture.modelling.page.html`.

Let us reformulate the task of clustering as follows: We will treat the vectors of expected values as cluster centres (rows of the M matrix). Instead of the cluster membership matrix U, we will consider the matrix $P = [p_{ij}]_{m \times k}$, where $p_{ij} = p(C_j|\mathbf{x}_i)$ means the probability that the i-th object belongs to the j-th cluster. Now let us define the target function for clustering as follows:

$$J_{EM}(P, M) = -\sum_{i=1}^{n} \log \Big(\sum_{j=1}^{k} p(\mathbf{x}_i|C_j)p(C_j) \Big) \tag{3.44}$$

The function $\log(\cdot)$ was introduced to simplify the computations. The minus sign allows to treat the problem of parameter estimation as the task of minimization of the expression (3.44).

Assuming that the set of observations consists really of k clusters we can imagine that each observation $\mathbf{x}_i, i = 1, \ldots, m$ was generated by exactly one of the k distributions, though we do not know by which. Hence, the feature that we shall subsequently call "*class*" is a feature with unknown values [17]. We can say that we have incomplete data of the form $\mathbf{y} = (\mathbf{x}, class)$. Further, we do not know the parameters of the constituent distributions of the mixture (3.42). In order to estimate them, we can use the EM (*Expectation Maximization*) method, elaborated by Dempster, Laird and Rubin [94], which is, in fact, a maximum likelihood method, adapted to the situation when data is incomplete. The method consists in repeating the sequence of two steps (see [247], [55]):

- Step **E** (*Expectation*): known feature values of objects, together with estimated parameters of models are used to compute for these objects the expected values of the unknown ones.
- Step **M** (*Maximization*): both known (observed) and estimated (unobserved) feature values of objects are used to estimate the model parameters using the model likelihood maximisation given the data.

The EM algorithm is one more example of the broad class of hill-climbing algorithms. Its execution starts with a choice of mixture parameters and proceeds by stepwise parameter re-estimation until the function (3.44) reaches a local optimum.

[17] More precisely, it is a so called "hidden feature" or "hidden variable".

Step **E** consists in estimation of the probability that the i-th object belongs to the class j. If we assume that we know the parameters of distributions belonging to the mixture (3.42) then we can estimate this probability using the Bayes theorem:

$$p(C_j|\mathbf{x}_i) = \frac{p(\mathbf{x}_i|C_j)p(C_j)}{p(\mathbf{x}_i)} = \frac{p(\mathbf{x}_i|C_j)p(C_j)}{\sum_{l=1}^{k} p(\mathbf{x}_i|C_l)p(C_l)} \tag{3.45}$$

$p(\mathbf{x}_i|C_j)$ is the value of the density function $\phi(\mathbf{x}_i; \boldsymbol{\mu}, \Sigma)$, having mean vector $\boldsymbol{\mu}$ and covariance matrix Σ, at the point $\mathbf{x}_i$, i.e. $p(\mathbf{x}_i|C_j) = \phi(\mathbf{x}_i; \boldsymbol{\mu}_j, \Sigma_j)$. The value $p(\mathbf{x}_i)$ is computed according to the total probability theorem.

To execute step **M**, we assume that we know the class membership of all objects, which is described by the aposteriorical distribution $p(C_j|\mathbf{x}_i)$. Under these circumstances the method of likelihood maximisation can be used to estimate the distribution parameters. The estimator of the vector of expected values of the j-th cluster is of the form:

$$\boldsymbol{\mu}_j = \frac{1}{mp(C_i)} \sum_{i=1}^{m} p(C_j|\mathbf{x}_i)\mathbf{x}_i \tag{3.46}$$

Note that this equation has the same form as the equation (2.42) given that the weights $w(\mathbf{x}_i)$ are constant, e.g. $w(\mathbf{x}_i) = 1$. If we assume, additionally, the preferential class membership (i.e. $p(C_j|\mathbf{x}_i) \in \{0, 1\}$) then we will discover that the formula (3.46) can be reduced to the classical formula of computation of gravity centre of a class.

Similarly, we compute the covariance matrix from the formula:

$$\Sigma_j = \frac{1}{mnp(C_j)} \sum_{i=1}^{m} (\mathbf{x}_i - \boldsymbol{\mu}_j)(\mathbf{x}_i - \boldsymbol{\mu}_j)^{\mathsf{T}} \tag{3.47}$$

and the apriorical class membership probability from the equation:

$$p(C_j) = \frac{1}{m} \sum_{i=1}^{m} p(C_j|\mathbf{x}_i) \tag{3.48}$$

We can summarize these remarks with the algorithm 3.9.

If we speak in the language of the k-means algorithm, then we will say that the step **E** is equivalent to an update of the assignment of points to clusters, and the step **M** can be deemed as determining characteristics of the new partition of the set X. A convenient approach (see e.g. [87]) to implement the above-mentioned pseudo-code consists in updating apriorical probabilities right after step **E**, thereafter determining centres $\boldsymbol{\mu}_i^{t+1}$, and lastly computing the matrix Σ_j^{t+1}.

System `WEKA` [363] implements the EM algorithm for cluster analysis. Many other systems exploit mixtures of distributions, e.g. `Snob` [357], `AutoClass` [70] and `MClust` [131].

Algorithm 3.9 EM algorithm for cluster analysis

```
1: Initialisation. Set the iteration counter to $t = 0$. Give the initial estimates of the parameters of the distributions $\boldsymbol{\mu}_j^t, \Sigma_j^t$ and of apriorical values of probabilities $p(C_j)$, $j = 1, \dots, k$.
2: (Step E): Using Bayes Theorem compute the membership of $i$-th object in $j$-th class, $p^{t+1}(C_j|\mathbf{x}_i), i = 1, \dots, m, j = 1, \dots, k$ – compare equation (3.45).
3: (Step M): Knowing class memberships of objects update the distribution parameters $\boldsymbol{\mu}_i^{t+1}, \Sigma_j^{t+1}$, and apriorical probabilities $p^{t+1}(C_j), j = 1, \dots, k$ – equations (3.46) – (3.48).
4: $t = t + 1$
5: Repeat steps 2 – 4 till the estimate values stabilise.
```

Experiments presented in [159] show that the EM algorithm is, with respect to quality, comparable to k-means algorithm. Just like in the case of the latter, also performance of the EM algorithm depends on initialisation[18]. However, as noticed in [7], the EM algorithm is not suitable for processing of high-dimensional data due to losses in computational precision. Let us remember, too, that determining the covariance matrix requires computation of $\frac{1}{2}(n^2 + n)$ elements, which becomes quite expensive with the growth of the number of dimensions. Finally, the disadvantage of the EM algorithm formulated in the here presented matter is its slow convergence, [94]. But, on the other hand, it is much more universal than the k-means algorithm: it can be applied in the analysis of non-numerical data. The quick algorithm developed by Moore in [261] allows to reduce, at least partially, the deficiencies listed.

Structural similarity between the EM and k-means algorithms permits to state that if the distribution $p(C_j|\mathbf{x})$ is approximated by the rule "the winner takes all", then the EM algorithm with a mixture of Gaussian distributions with covariance matrix $\Sigma_j = \epsilon\mathbb{I}, j = 1, \dots, k$, where $\mathbb{I}$ means a unit matrix, becomes the k-means algorithm when $\epsilon \to 0$ [202].

Dasgupta and Schulman [87] analyse deeply the case of a mixture of spherical normal distributions, that is- those with $\Sigma_j = \sigma_j\mathbb{I}$. They prove that if the clusters are sufficiently well separable, see equation (2.56), then the estimation of mixture parameters requires only two iterations of a slightly modified algorithm 3.9. This is particularly important in case of high-dimensional data ($n >> \ln k$). They developed also a simple procedure, applicable in the cases when the exact value of k is not known in advance. They propose an initialisation method, which is a variant of the method (c) from Section 3.1.3.

In recent years, we heve been witnessing a significant advancement of theoretical description and understanding of the problems related to learning of parameters of a mixture of distributions. A Reader interested in these aspects is recommended to study the paper [194]. We will return to this topic in Section 5.2.5.4.

[18] One of initialisation methods for the EM algorithm, estimating mixture parameters, consists in applying the k-means algorithm.

Slow convergence of the standard version of the EM algorithm urged the development of its various modifications, such as greedy EM[19], stochastic EM[20], or random swap EM. Their brief review can be found in sections 4.4. i 4.5 of the PhD thesis [388] and in paper [284], where a genetic variant of the EM algorithm has been proposed. It allows not only to discover the mixture parameters but also the number of constituent components.

3.3 FCM: fuzzy *c*-means algorithm

3.3.1 Basic formulation

Ruspini [297] was the first to propose to represent clusters as fuzzy sets. If $\widetilde{\chi}_j$ denotes the membership function representing j-th cluster then the fuzzy partition is the set of fuzzy subsets $\widetilde{F}_1, \dots, \widetilde{F}_k$. Their membership functions are defined as follows:

$$\sum_{j=1}^{k} \widetilde{\chi}_j(\mathbf{x}_i) = 1, i = 1, \dots, m \tag{3.49}$$

This definition is intended to take into account *outliers* and other irregularities in the induced partition. Additionally, to be able to determine representatives of individual clusters (called prototypes or – as in the case of crisp sets – gravity centres), Dunn [114] introduced the concept of compact and well separated (CWS) clusters. Let $\mathcal{C} = \{C_1, \dots, C_k\}$ mean a crisp partition of a set of objects. Let $(\mathbf{x}, \mathbf{y}) \in X$ be two such points that $\mathbf{x} \in C_i$, $\mathbf{y} \in conv(C_i)$ where $conv(C_i)$ is a convex hull C_i, and let $i, j, k, j \neq k$ be any indexes. The set C_i is a CWS-cluster if $(\mathbf{x}, \mathbf{y})$ are much closer to one another in the sense of some assumed distance d than any two points $(\mathbf{u}, \mathbf{v}) \in X$ such that $\mathbf{u} \in C_j, \mathbf{v} \in conv(C_k)$. This property is described by the index

$$\beta(k, \mathcal{C}) = \Big(\min_{j=1,\dots,k} \quad \min_{l=1,\dots,m, l\neq j} d(C_j, conv(C_l)) \Big) / \max_{j=1,\dots,k} diam(C_j) \tag{3.50}$$

where $diam(C_j)$ is the diameter of the set C_j, and $d(A, B)$ is the distance between the sets A and B. It turns out that the set X can be divided into k CWS-clusters (when the distance d is known), if [114]

$$\overline{\beta}(k) = \max_{\mathcal{C}} \beta(k, \mathcal{C}) > 1 \tag{3.51}$$

The problem of finding a partition, for which the index $\beta(k, \mathcal{C}) = \overline{\beta}(k)$, is very difficult. In particular, Dunn [114] has shown that the algorithm `ISODATA`,

[19] See J.J. Verbeek, N. Vlassis, B. Kröse: "Efficient greedy learning of Gaussian mixture models", *Neural computation*, **15**(2), 2003, pp. 469-485.

[20] G. Celeux, D. Chauveau, J. Diebolt: Stochastic versions of the EM algorithm: an experimental study in the mixture case. *J. of Stat. Computation and Simulation*, **55**(4), 1996, pp. 287-314.

introduced in Section 3.1.4, finds the CWS-partition also in cases when it does not exist. For this reason the criterion (3.1) was weakened allowing for fuzzy partitions of the set of objects. Let $U = [u_{ij}]_{m\times k}$ be a matrix fulfilling the following conditions

$$\begin{aligned}
&(a)\ 0 \leq u_{ij} \leq 1, i = 1,\dots,m, j = 1,\dots,k \\
&(b)\ \sum_{j=1}^{k} u_{ij} = 1, i = 1,\dots,m \\
&(c)\ 0 < \sum_{i=1}^{m} u_{ij} < m, j = 1,\dots,k
\end{aligned} \tag{3.52}$$

Element u_{ij} determines the membership degree of the object $\mathbf{x}_i$ in the class C_j. The condition (a) means a partial membership of objects the classes, and condition (b) enforces the full membership of objects in a distinguished set of classes[21]. Finally, the condition (c) does not permit to create empty classes. The set of all matrices fulfilling the conditions (3.52) is called the set of fuzzy partitions and we denote it with the symbol $\mathcal{U}_{fk}$. Let us make the remark that the dimension of the space of fuzzy clusterings significantly exceeds the dimension of the space of crisp partitions [54].

Prototypes of classes are represented by the rows of the matrix $M = (\boldsymbol{\mu}_1,\dots,\boldsymbol{\mu}_k)^{\mathrm{T}}$ having the dimensions $k \times n$.

The search for a fuzzy partition fulfilling the conditions mentioned is carried out via minimisation of the quality index of the form:

$$J_\alpha(U,M) = \sum_{i=1}^{m}\sum_{j=1}^{k} u_{ij}^{\alpha}\|\mathbf{x}_i - \boldsymbol{\mu}_j\|^2 \tag{3.53}$$

where $\alpha > 1$ is a parameter (so-called fuzziness exponent). It has been shown [54] that when $\alpha \to 1$ then the algorithm generates clusters identical with ones obtained from the algorithm `ISODATA` (which justifies the name *Fuzzy ISODATA* for the algorithm described here), and when $\alpha \to \infty$ then the values $u_{ij} \to 1/k$, i.e. we obtain a completely fuzzy partition. Typical values of the parameter α are 1.25 and 2.

Introduction of partial object membership allows to distinguish between the "typical" group representatives and the objects that are less likely characterised as belonging to a given group. The idea has been illustrated in Figure 3.7.

Remark 3.3.1 *Due to similarity of the function (3.53) to (3.1), the subsequently discussed algorithm could be called fuzzy k-means algorithm. Its authors denoted the number of clusters with the letter c, hence they coined the name Fuzzy c-Means. We will, however, continue to denote the number of clusters with the letter k, while keeping the abbreviation FCM when referring to this algorithm.* □

[21] In other words, we do not allow for the existence of other (unknown) classes, to which objects could partially or completely belong

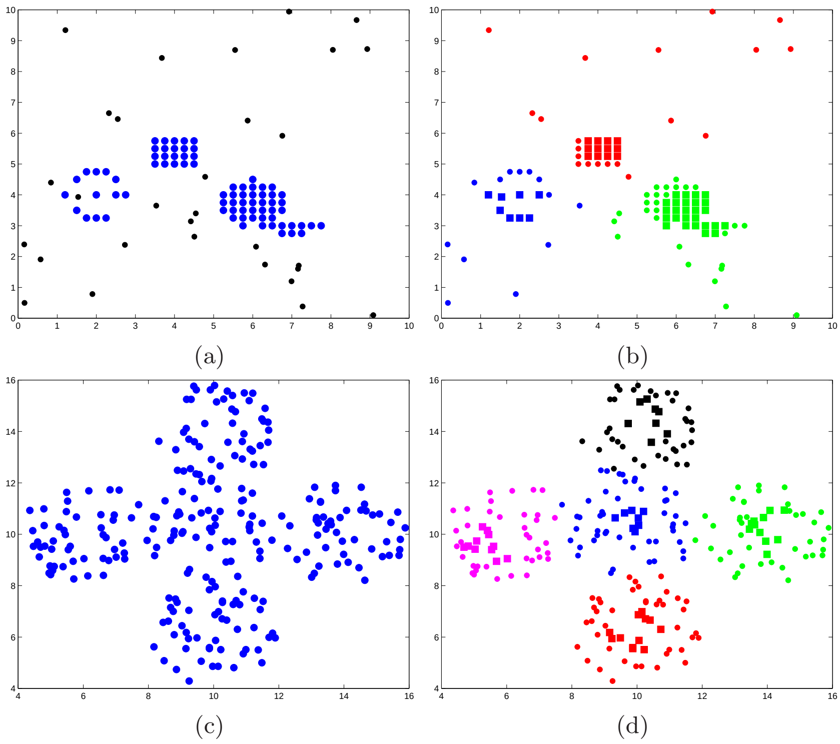


Fig. 3.7. Clustering of data with the use of the FCM algorithm: (a) Set `data3_2` (blue points) with added noise (black points). (b) Assignments to groups generated by the FCM algorithm. The bigger squares denote typical objects, smaller circles – the other objects; the colour indicates the class membership. Same convention applies to Figure (d) representing the results obtained for the set `data5_2` from Figure (c). $\alpha = 2$ was assumed in computations. Object $\mathbf{x}_i$ is treated as typical if $\max_j u_{ij} \geq 0.9$.

Remark 3.3.2 *The Euclidean distance occurring in the equation (3.53) can be replaced with any norm* $\|\mathbf{x}\|_W = (\mathbf{x}^T W \mathbf{x})^{1/2}$*, where* W *is a positive definite matrix of dimension* $n \times n$*. This more general formulation has been used, e.g., in the book [51]. Subsequent formulas for the membership degrees* u_{ij} *remain valid if the Euclidean distance used therein is replaced by the norm* $\|\cdot\|_W$*.* □

3.3.2 Basic FCM algorithm

The indicator (3.53), depending on the partition U and the prototypes M, is not a convex function of its arguments. Nonetheless, the problem of its minimisation can be simplified by fixing the values of either matrix U or M, that is, by assuming $U = \widetilde{U}$, $M = \widetilde{M}$, respectively. In such a case the simplified functions $\widetilde{J}_m(U) = J_m(U, \widehat{M})$ and $\widetilde{J}_m(M) = J_m(\widehat{U}, M)$ are convex functions of their arguments – consult [50], [189]. Hence, the classical optimisation methods are applicable, i.e. to determine the vector of prototypes, the equation system of k equations of the form given below, is solved

$$\frac{\partial}{\partial \boldsymbol{\mu}_j} J_\alpha(U, M) = 0, \quad j = 1, \ldots, k \tag{3.54}$$

and solving for the matrix U relies on creating the Lagrange function

$$L(J_\alpha, \lambda) = \sum_{i=1}^{m} \sum_{j=1}^{k} u_{ij}^\alpha \|\mathbf{x}_i, \boldsymbol{\mu}_j\|^2 - \sum_{i=1}^{m} \lambda_i (\sum_{j=1}^{k} u_{ij} - 1) \tag{3.55}$$

and computing values u_{ij} from the equation system

$$\begin{cases} \frac{\partial}{\partial u_{ij}} L(J_\alpha, \lambda) = 0 \\ & i = 1, \ldots, m, \quad j = 1, \ldots, k \\ \frac{\partial}{\partial \lambda_i} L(J_\alpha, \lambda) = 0 \end{cases} \tag{3.56}$$

Solution of the task of minimisation of function (3.53) in the set of fuzzy partitions $\mathcal{U}_{fk}$ is obtained iteratively by computing, for a fixed partition U, the prototypes (see appendix A)

$$\mu_{jl} = \frac{\sum_{i=1}^{m} u_{ij}^\alpha \mathbf{x}_{il}}{\sum_{l=1}^{m} u_{lj}^\alpha}, \; j = 1, \ldots, k, l = 1, \ldots, n \tag{3.57}$$

and then new assignments to clusters

$$u_{ij} = \begin{cases} \left[\sum_{l=1}^{k} \left(\frac{\|\mathbf{x}_i - \boldsymbol{\mu}_j\|}{\|\mathbf{x}_i - \boldsymbol{\mu}_l\|} \right)^{\frac{2}{\alpha-1}} \right]^{-1} & \text{if } Z_i = \emptyset \\ \epsilon_{ij} & \text{if } j \in Z_i \neq \emptyset \\ 0 & \text{if } j \notin Z_i \neq \emptyset \end{cases} \tag{3.58}$$

where $Z_i = \{j \colon 1 \leq j \leq k, \|\mathbf{x}_i - \boldsymbol{\mu}_j\| = 0\}$, and values ϵ_{ij} are chosen in such a way that $\sum_{j \in Z_i} \epsilon_{ij} = 1$. Usually, the non-empty set Z_i contains one element, hence $\epsilon_{ij} = 1$. if we substitute $d(\mathbf{x_i}, \boldsymbol{\mu}_j) = \|\mathbf{x_i} - \boldsymbol{\mu}_j\| + \varepsilon$, where ε is a number close to zero, e.g. $\varepsilon = 10^{-10}$, then the above equation can be simplified to

$$u_{ij} = \left[\sum_{l=1}^{k} \left(\frac{d(\mathbf{x}_i, \boldsymbol{\mu}_j)}{d(\mathbf{x}_i, \boldsymbol{\mu}_l)} \right)^{\frac{2}{\alpha-1}} \right]^{-1} \tag{3.59}$$

This last equation can be rewritten in an equivalent form that requires a lower number of divisions

$$u_{ij} = \frac{d^{\frac{2}{1-\alpha}}(\mathbf{x}_i, \boldsymbol{\mu}_j)}{\sum_{l=1}^{k} d^{\frac{2}{1-\alpha}}(\mathbf{x}_i, \boldsymbol{\mu}_l)} \tag{3.60}$$

Note that the membership degree u_{ij} depends not only on the distances of the i-th object from the centre of the j-th cluster but also on its distance to centres of other clusters. Furthermore, when $\alpha = 2$, the denominator of the expression (3.60) is the harmonic average of the squares of distance of the object from the cluster centres. In this case, the FCM resembles a little bit the KHM algorithm from Section 3.1.5.4.

The FCM algorithm termination condition is usually defined as stabilisation of the partition matrix. If U^t, U^{t+1} denote the matrices obtained in subsequent iterations, then the computations are terminated as soon as $\max_{i,j} |u_{ij}^{t+1} - u_{ij}^t| \leq \epsilon$, where ϵ is a predefined precision, e.g. $\epsilon = 0.0001$. Of course, one can change the order of steps, i.e. first initiate the prototype matrix M and determine the corresponding cluster assignments u_{ij}, and subsequently update the prototypes. The last two steps are repeated till the vectors stabilise $\boldsymbol{\mu}_j$, i.e. till $|\mu_{jl}^{t+1} - \mu_{jl}^t| < \epsilon$, where $j = 1, \dots, k$, $l = 1, \dots, n$. Note that the number of comparisons required in deciding on computation termination is in the second case usually lower than in the first one: matrix M contains kn elements, while matrix U has only mk elements.

The fuzziness coefficient α is an important parameter of the algorithm, because its properties heavily depend on the value od this coefficient. If this value is close to one, the algorithm behaves like the classic k-means algorithm. If α grows without bounds, then prototypes converge to the gravity centre of the object set X. Several heuristic methods for selection of this coefficient have been proposed in the literature [45], [275]. The best recommendation is to choose it from the interval $[1.5, 2.5]$. One has, however, to remember that the mentioned heuristics result from empirical investigations and may not reflect all the issues present in real world data. The paper [377] suggests some rules for the choice of the value of α, pointing at the fact that coefficient choice depends on the data themselves. The impact of parameter choice on the behaviour of the FCM algorithm was investigated by Choe and Jordan in [74]. The influence of parameter α on the number of iterations untill stabilisation of the matrix U and on the distance of returned prototypes $\boldsymbol{\mu}_j$ from the intrinsic ones $\boldsymbol{\mu}_j^*$ is depicted in Figure 3.8. The average distance has been computed as

$$d_{avg} = \frac{1}{k} \sum_{j=1}^{k} \|\boldsymbol{\mu}_j^* - \boldsymbol{\mu}_j\| \tag{3.61}$$

We chose the set `iris.data` stemming from the repository [25] to illustrate the comparison. Intrinsic centres of the groups are presented in the table below.

$$\mathbf{M}^* = \begin{bmatrix} 5.00 & 3.42 & 1.46 & 0.24 \\ 5.93 & 2.77 & 4.26 & 1.32 \\ 6.58 & 2.97 & 5.55 & 2.02 \end{bmatrix} \tag{3.62}$$

We can deduce from the figure that the increase of the value of the coefficient α causes the increase in the number of iterations. Interestingly, for α values close to 1 the error measured with the quantity d_{avg} initially decreases and then, after exceeding the optimal value (here $\alpha^* = 1.5$), it grows. The standard deviation is close to zero for low α values, which means a high repeatability of the results.

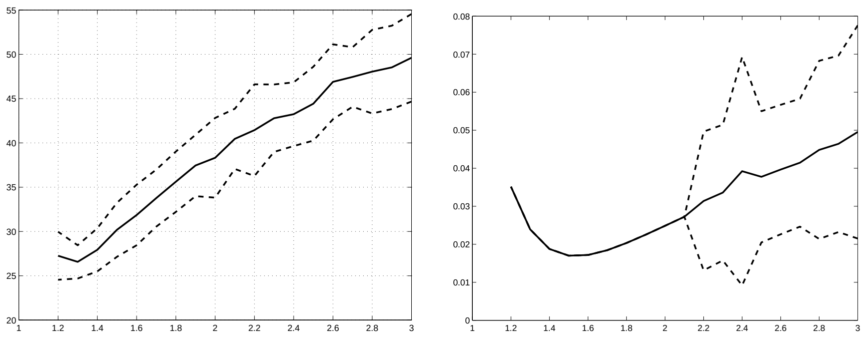


Fig. 3.8. Influence of the value of fuzziness exponent α on the number of iterations (picture to the left) and the quality of prototypes (picture to the right), expressed in terms of the average distance d_{avg} according to the equation (3.61). Solid line indicates the mean value $\overline{m}$ of the respective quantity (averaged over 100 runs), and the dotted lines mark the values $\overline{m} \pm \overline{s}$, with $\overline{s}$ being the standard deviation. $\epsilon = 10^{-8}$ was assumed. Experiments were performed for the data set `iris.txt`.

Remark 3.3.3 *Schwämmle and Jensen [308] investigated randomly generated data sets and concluded that the best value of the α parameter is a function of the dimension n and the amount of data m. Based on their experiments, they proposed the following analytical form of this function*

$$\begin{aligned}\alpha(m,n) = 1 &+ \left(\tfrac{1418}{m} + 22.05\right)n^{-2} \\ &+ \left(\tfrac{12.33}{m} + 0.243\right)n^{-0.0406\ln(m)-0.1134}\end{aligned} \tag{3.63}$$

This result contradicts the common practice to choose $\alpha = 2$.

It has been also observed that investigation of the minimal distance between the cluster centres, V_{MCD}, is a good hint for choice of the proper number of clusters k. It is recommended to choose as a potential candidate such a value k^* for which an abrupt decrease of the value of V_{MCD} occurs.

One should, however, keep in mind that these results were obtained mainly for the Gaussian type of clusters. Both authors suggest great care when applying the derived conclusions to the real data. □

The FCM algorithm converges usually quickly to a stationary point. Slow convergence should be rather treated as an indication of a poor starting point. Hathaway and Bezdek cite in [165, p. 243] experimental results on splitting a mixture of normal distributions into constituent sets. An FCM algorithm with parameter $\alpha = 2$ needed 10-20 iterations to complete the task, while the EM algorithm required hundreds, and in some cases thousands of iterations.

Encouraged by this discovery, the authors posed the question: Let $p(\mathbf{y}; \alpha_0, \alpha_1) = \alpha_0 p_0(\mathbf{y}) + \alpha_1 p_1(\mathbf{y})$, where p_0 i p_1 are symmetric density functions with mean values 0 and 1, respectively, and the expected values (with respect to components) of the variable $|Y|^2$ being finite. Can these clusters be identified? Regrettably, it turns out that (for $\alpha = 2$) there exist such values of

α_0, α_1 that the FCM algorithm erroneously identifies the means of both sub-populations. This implies that even if one could observe an infinite number of objects the algorithm possesses only finite precision (with respect to estimation of prototype location). This result is far from being a surprise. FCM is an example of non-parametric algorithm and the quality index J_α does not refer to any statistical properties of the population. Hathaway and Bezdek conclude, [165]: if population components (components of a mixture of distributions) are sufficiently separable, i.e. each sub-population is related to a clear "peak" in the density function, then the FCM algorithm can be expected to identify the characteristic prototypes at least as well as the maximum likelihood method (and for sure much quicker than this).

An initial analysis of the convergence of the FCM algorithm was presented in the paper [50], and a correct version of the proof of convergence was presented 6 years later in the paper [169]. A careful analysis of the properties of this algorithm was initialised by Ismail and Selim [189]. This research direction was pursued, among others, by Kim, Bezdek and Hathaway [204], Wei and Mendel [361], and also Yu and Yang [378].

3.3.3 Measures of quality of fuzzy partition

Though the FCM algorithm requires only a small number of iteration till a stable partition of objects is obtained, the partition quality assessment requires the introduction of appropriate quality measures that would depend on the parameters k and α. The reader will find a thorough discussion of clustering quality evaluation methods in chapter 4. Nonetheless, we will present here a couple of measures that play a special role in the evaluation of fuzzy clustering algorithms.

If the intrinsic group membership of objects is known, $\mathcal{P}^t = \{C_1^t, \dots, C_k^t\}$, then the so-called purity index is applied. It reflects the agreement between the intrinsic partition and the one found by the algorithm. First, the fuzzy assignment matrix is transformed to a Boolean group membership matrix U^b with entries

$$u_{ij}^b = \begin{cases} 1 \text{ if } j = \arg\min\limits_{1 \le t \le k} u_{it} \\ 0 \text{ otherwise} \end{cases} \tag{3.64}$$

A partition $\mathcal{P}^f = \{C_1^f, \dots, C_k^f\}$ is obtained in this way, with $C_j^f = \{i \colon u_{ij}^b = 1\}$. Subsequently, we construct the contingency table with entries $m_{ij} = |C_i^t \cap C_j^f|$. Finally, the agreement of the two partitions is calculated as

$$\mathfrak{P}(\mathcal{P}^1, \mathcal{P}^2) = \frac{1}{m} \sum_{i=1}^{k_1} \max_{1 \le j \le k_2} m_{ij} \tag{3.65}$$

While the purity is a general purpose measure, the so called reconstruction error is a measure designed exclusively for algorithms producing fuzzy partitions [280]. It is defined as the average distance between the original object and the reconstructed one, i.e.

$$e_r = \frac{1}{m}\sum_{i=1}^{m} \|\mathbf{x}_i - \widetilde{\mathbf{x}}_i\|^2 \tag{3.66}$$

where the reconstruction $\widetilde{\mathbf{x}}_i$ is performed according to the formula

$$\widetilde{\mathbf{x}}_i = \frac{\sum_{j=1}^{k} u_{ij}^{\alpha} \boldsymbol{\mu}_j}{\sum_{j=1}^{k} u_{ij}^{\alpha}}, \qquad i = 1, \ldots, m \tag{3.67}$$

The lower the reconstruction error the better the algorithm performance. One should, however, remember that low values of the k parameter induce rather high error values ($e_r \to 0$ when $k \to m$). The purity index evaluates, therefore, the algorithm precision, while the reconstruction error describes the quality of encoding/decoding of objects by the prototypes and the assignment matrix. One can say that e_r is a measure of dispersion of prototypes in the feature space. In particular, the reconstruction error decreases when the prototypes are moved towards the centres of dense areas of feature space [147]. Hence, e_r measures the capability of prototypes to represent individual clusters. The dependence of the reconstruction error on the fuzziness exponent α is illustrated in Figure 3.3.3.

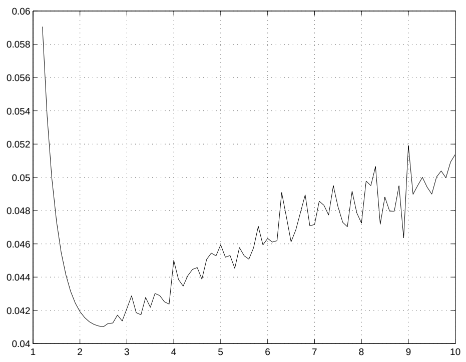


Fig. 3.9. Influence of the α exponent (abscissa axis) on the reconstruction error (ordinate axis). Test data: file `iris.txt`.

Other measures of quality of fuzzy partitions are partition coefficient $F_k(U)$ and fuzzy partition entropy $H_k(U)$. They are defined as follows, see e.g. [51]:

$$F_k(U) = \mathrm{tr}(UU^{\mathrm{T}})/m = \frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{k} u_{ij}^2 \tag{3.68}$$

$$H_k(U) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{k} u_{ij}\log_a u_{ij}, \qquad a > 1 \tag{3.69}$$

They exhibit the following properties

$$
\begin{aligned}
&F_k(u) = 1 \Leftrightarrow H_k(U) = 0 \Leftrightarrow U \text{ is a crisp partition} \\
&F_k(U) = \tfrac{1}{k} \Leftrightarrow H_k(U) = \log_a(k) \Leftrightarrow U = [\tfrac{1}{k}] \qquad (3.70) \\
&\tfrac{1}{k} \le F_k(U) \le 1; \qquad 0 \le H_k(U) \le \log_a(k)
\end{aligned}
$$

The entropy H_k is more sensitive to local changes in partition quality than the coefficient F.

When data tend to concentrate into a low number of well separable groups, then these indicators constitute a good hint for the proper selection of the number of clusters.

Remark 3.3.4 *The quantity $H_k(U)$ is a measure indicating the degree of fuzziness of a partition. If U is a crisp partition, then $H_k(U) = 0$ for any matrix U with elements $u_{ij} \in \{0,1\}$. The entropy $H(U)$, defined later in equation (4.29) in Section 4.4.2, allows to explore deeper the nature of a crisp partition. One should not confuse these two measures.* □

Example 3.3.1 *To illustrate the above-mentioned thesis, consider two sets presented in Figure 3.10. The first set* `data_6_2` *contains two dimensional data forming six clear-cut clusters. The second set* `data_4_2` *was obtained by randomly picking same number of points from a two dimensional normal distribution $N(m_i, \mathbb{I})$, $i = 1, \dots, 4$, where $m_1 = (3,0)^T$, $m_2 = (0,3)^T$, $m_3 = (3,3)^T$, $m_4 = (0,6)^T$, and $\mathbb{I}$ means a unit covariance matrix. In both cases $F_k(U)$ and $H_k(U)$ for various values of k were computed, see Figure 3.11. In the first case of a clear structure, we observe clear cut optima reached by both indexes. In the second case both indices behave monotonically.* □

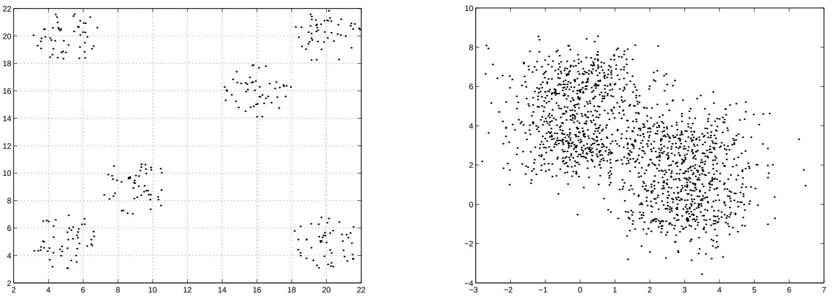


Fig. 3.10. Test sets `data_6_2` (Figure to the left) and `data_4_2` (Figure to the right)

3.3.4 An alternative formulation

The method of determining the membership degree via equation (3.59) does not depend on the definition of distance, [51]. By replacing the Euclidean distance $\|\mathbf{x}_i - \boldsymbol{\mu}_j\|$ used there with some distance $d(\mathbf{x}_i, \boldsymbol{\mu}_j)$, we can state that

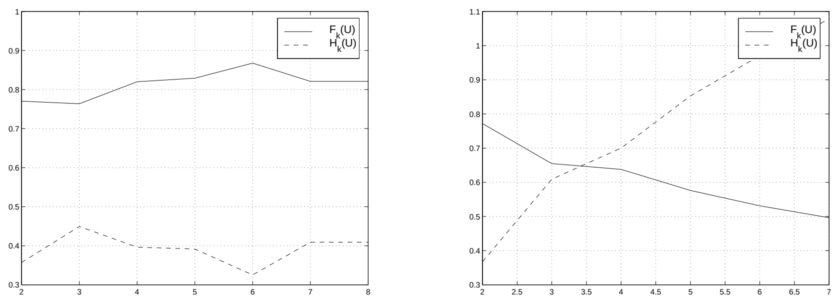


Fig. 3.11. Dependence of the values of quality criteria for the sets `data_6_2` (to the left) and `data_4_2` (to the right) on the assumed number of classes

$$u_{ij}^{\alpha-1} d^2(\mathbf{x}_i, \boldsymbol{\mu}_j) = \left(\frac{1}{\sum_{l=1}^{k} d^{\frac{2}{1-\alpha}}(\mathbf{x}_i, \boldsymbol{\mu}_l)} \right)^{\alpha-1} = \left(\sum_{l=1}^{k} d^{\frac{2}{1-\alpha}}(\mathbf{x}_i, \boldsymbol{\mu}_l) \right)^{1-\alpha}$$

Hence

$$\begin{aligned}
\widetilde{J}_\alpha(M) &= \sum_{i=1}^{m} \sum_{j=1}^{k} u_{ij} u_{ij}^{\alpha-1} d^2(\mathbf{x}_i, \boldsymbol{\mu}_j) \\
&= \sum_{i=1}^{m} \sum_{j=1}^{k} u_{ij} \left(\sum_{l=1}^{k} d^{\frac{2}{1-\alpha}}(\mathbf{x}_i, \boldsymbol{\mu}_l) \right)^{1-\alpha} \\
&= \sum_{i=1}^{m} \left(\sum_{l=1}^{k} d^{\frac{2}{1-\alpha}}(\mathbf{x}_i, \boldsymbol{\mu}_l) \right)^{\alpha-1} \cdot \sum_{j=1}^{k} u_{ij}
\end{aligned}$$

The above equation allows to replace the criterion function (3.53) with an equivalent function (see e.g. [166], [361])

$$\widetilde{J}_\alpha(M) = \sum_{i=1}^{m} \Big(\sum_{j=1}^{k} d^{\frac{2}{1-\alpha}}(\mathbf{x}_i, \boldsymbol{\mu}_j) \Big)^{1-\alpha} \tag{3.71}$$

More precisely, if the distances $d(\mathbf{x}_i, \boldsymbol{\mu}_j)$ are continuous functions of parameters $M \in \mathcal{M}$, which describe the group prototypes, and $\mathcal{M}$ is an open subset $\mathbb{R}^{kn}$, and for $M^* \in \mathcal{M}$ all distances $d(\mathbf{x}_i, \boldsymbol{\mu}_j) > 0$, $i = 1, \dots, m$, $j = 1, \dots, k$, then, [166]

(a) If (U^*, M^*) is a global (resp. local) minimum of the index $J_\alpha(U, M)$ in $\mathcal{U}_{kc} \times \mathcal{M}$ then M^* is a global (resp. local) minimum of index $\widetilde{J}_\alpha(M)$ in the set $\mathcal{M}$.
(b) If M^* is a global (resp. local) minimum of the index $\widetilde{J}_\alpha$ in the set $\mathcal{M}$ then the pair $(\Phi(M^*), M^*)$ is a global (resp. local) minimum of the index J_α in

the set $\mathcal{U}_{fk} \times \mathcal{M}$, where $\Phi(M)$ is a mapping assigning the given matrix M a respective assignment matrix U.

The difference between the formulations (3.53) and (3.71) is not only formal. Let us notice that in the first case the number of parameters to be identified is $k(m+n)$, and in the second case there are kn of them.

More importantly, in the first case we deal with an optimisation task with constraints (3.60). The solution of the problem stated in this way is the alternating algorithm from Section 3.3.2.

In the second case, the determination of the allocations from the family $\mathcal{U}_{fk}$ is ignored, with concentration on finding such prototypes that minimise the indicator (3.71). Here we deal with a much simpler task of unconditional optimisation. Class allocations for objects are computed from equation (3.60) only if needed.

To illustrate more convincingly the difference between these approaches, let us assume that we replace the Euclidean distance used so far with a distance induced by Minkowski metric [22]. This requires a modification of the iterative algorithm in the formulation (3.53). Such changes are unnecessary in the optimisation driven formulation (3.71). Only the value of the indicator $\widetilde{J}_\alpha(M)$ is computed in a slightly different manner.

An obvious requirement is to dispose of an efficient optimisation procedure. If the number of features is not big, then the simplex algorithm (called also creeping amoeba) can be applied. The algorithm is authored by Nelder and Mead – compare chapter 10 w [287]. Application of a genetic algorithm to optimise the indicator (3.71) was presented in [158].

More details about the equivalence of solutions obtained in both cases are presented in paper [166].

3.3.5 Modifications of the FCM algorithm

The base FCM algorithm assumes that the distance between an object and a prototype is an Euclidean one. But this is not the best way of measuring the similarity of the multidimensional objects to distinguished prototypes (see Section 2.2). Below we present several different modifications of the target function having the general form

$$J_\alpha(U,M) = \sum_{j=1}^{k}\sum_{i=1}^{m} u_{ij}^{m} d^2(\mathbf{x}_i, \boldsymbol{\mu}_j) \tag{3.72}$$

In Section 3.3.5.1 in place of $d(\mathbf{x}_i, \boldsymbol{\mu}_j)$ we consider Minkowski distance. The cluster allocation applied in FCM is sufficient if the clusters are Voronoi-shaped. Therefore, in Section 3.3.5.2 in place of $d(\mathbf{x}_i, \boldsymbol{\mu}_j)$ we assume Mahalanobis distance. It enables the discovery of clusters with varying shapes and densities. Another modification, adopted for such a situation was proposed by Gath and

[22] This problem is discussed in depth in Section 3.3.5.1.

Geva in [139]. If the shape of the clusters is known in advance (e.g. line segments, circles, ellipsoids), the algorithms presented in Sections 3.3.5.3 and 3.3.5.4 may prove useful. In Section 3.3.5.5 we present a generalisation of the spherical algorithm from Section 3.1.5.3. We will present also two kernel variants of the base FCM algorithm (Section 3.3.5.6).

Constraints imposed on matrix U (see equation (3.52) from page 99) cause that the algorithm is sensitive to the presence of non-typical data (*outliers*). Wu and Yang [366] attempted to overcome this shortcoming by suggesting the extension of the set of k groups with an additional cluster containing all non-typical data. Another idea is to give up the requirement that the sum of membership degrees to all groups be equal 1. This leads to so-called possibilistic clustering algorithm that we discuss in Section 3.3.5.7.

Finally, in Section 3.3.5.8 an algorithm is presented in which instead of distance, the matrix of dissimilarities R is used, in which entries quantify the dissimilarity between pairs of objects.

An exhaustive review of all FMC algorithm modifications is beyond the scope of this book. An interested reader is kindly advised to consult, e.g., the books [51], [54], [93], [177]. A systemat review of other modifications of the FCM algorithms, including the ones for observations stemming from a mixture of various probability distributions, is presented in [376].

The basic version of FCM requires storing the allocation matrix U, which increases its memory complexity and, in consequence, also time complexity. But with knowledge of the prototype vectors, elements of this matrix can be restored quite efficiently, if needed. Such a solution was presented by Kolen and Hutcheson in the paper [210]. Hore, Hall and Goldgof present in [178] the so-called weighted FCM algorithm, developed for very big data sets. Another big data analysis variant of the algorithm was proposed in paper [73].

Let us note at the end the interesting attempts to merge self-organising networks of Kohonen[23] with FCM algorithm. First attempts were presented in [186], and their refinement is the FKCN algorithm (*Fuzzy Kohonen Clustering Network*), described in papers [276] and [340].

3.3.5.1 FCM algorithm with Minkowski metric

As announced in Section 2.2, let us start with the most natural modification of the algorithm, consisting in replacement of the Euclidean distance d_2 with its generalisation, that is, with Minkowski distance d_p, $p > 0$, defined by the formula (2.3), in particular - the Manhattan distance d_1. This modification was deemed to make the algorithm resistant against outliers. So, in the general case, the function (3.72) can have the form (see also [167])

[23] T. Kohonen. *Self-Organization and Associative Memory*, Springer-Verlag, Berlin, Heidelberg, 1898.

$$
\begin{aligned}
J_{\alpha,p}(U,M) &= \sum_{i=1}^{m}\sum_{j=1}^{k} u_{ij}^{\alpha}\|\mathbf{x}_i - \boldsymbol{\mu}_j\|_p^p \\
&= \sum_{i=1}^{m}\sum_{j=1}^{k}\sum_{l=1}^{n} u_{ij}^{\alpha}|x_{il} - \mu_{jl}|^p
\end{aligned}
\tag{3.73}
$$

We will seek here a solution by alternating computation of the elements of the U and M matrices just like in the case of the base FCM algorithm. Elements of the U matrix are determined according to equation (3.58) except that the Euclidean distance used there is substituted with Minkowski distance $\|\mathbf{x}_i - \boldsymbol{\mu}_j\|$.

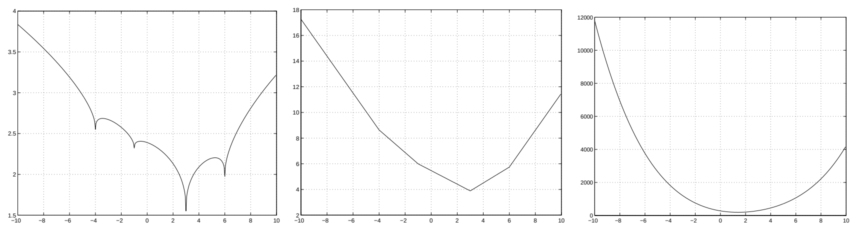
Determining components of prototypes is, however, more expensive than in the classical case of $p = 2$, because one has to find kn minima of one-dimensional functions of the form

$$
f_{jl}(\mu_{jl}) = \sum_{i=1}^{m} u_{ij}^{\alpha}|x_{il} - \mu_{jl}|^p,\ j = 1,\dots,k, l = 1,\dots,n \tag{3.74}
$$

A function defined in this way is not convex for $0 < p < 1$; its diagram has "spikes" at points $\mu_{jl} = x_{il}$, compare Figure 3.12(a). Determining the value μ_{jl}^* minimising the function f consists, therefore, in computing its values for all arguments x_{il}, $i = 1,\dots,m$ and choosing the proper one among them, that is

$$
\mu_{jl}^* = \arg\min_{1\le i\le m} f_{jl}(\mu_{jl}) \tag{3.75}
$$

We obtain a piece-wise linear function in case of $p = 1$, compare Figure 3.12(b), but the solution is found by the same procedure. Finally, in case of $p > 1$ we have to do with a convex function with a single minimum, Figure 3.12(c).



(a) $p = 0.4$ (b) $p = 1.0$ (c) $p = 3.5$

Fig. 3.12. Diagrams of the function (3.74) for three values of the parameter p. Function is of the form $f(x) = 0.6^{\alpha}|-4-x|^p + 0.5^{\alpha}|-1-x|^p + 0.8^{\alpha}|3-x|^p + 0.7^{m}|6-x|^p$, where x means μ_{jl}, and $\alpha = 2.5$.

Jajuga[24] presented an effective method of minimising function (3.73) for $p = 1$. By changing the summation order and by introducing weight

$$w_{ijl} = \frac{u_{ij}^{\alpha}}{|x_{il} - \mu_{jl}|} \tag{3.76}$$

the target function (3.73) can be represented in an equivalent form

$$J_{\alpha,1}(W, M) = \sum_{j=1}^{k} \sum_{l=1}^{n} \sum_{i=1}^{m} w_{ijl}(x_{il} - \mu_{jl})^2 \tag{3.77}$$

resembling the original target function (3.53) with parameter $\alpha = 1$. Cluster membership degrees can be determined in the following (already known) way

$$u_{ij} = \left(\sum_{l=1}^{k} \frac{d_{ij}^{1/(1-\alpha)}}{d_{il}^{1/(1-\alpha)}} \right)^{-1}, d_{ij} = \sum_{l=1}^{n} |x_{il} - \mu_{jl}| \tag{3.78}$$

But determining prototypes requires solving kn equations of the form

$$\min \sum_{i=1}^{m} w_{ijl}(x_{il} - \mu_{jl})^2, \quad j = 1, \dots, k, l = 1, \dots, n$$

i.e. knowing the actual values of u_{ij}, the weights are computed according to equation (3.76), and then prototype coordinates are calculated

$$\mu_{jl} = \frac{\sum_{i=1}^{m} w_{ijl} x_{il}}{\sum_{i=1}^{m} w_{ij}} \tag{3.79}$$

A review of other methods of fuzzy clustering for various values of the parameter p is contained in the paper [167]).

3.3.5.2 Gustafson-Kessel (GK) algorithm

Both FCM algorithm and its crisp counterpart prefer Voronoi ("spherical") clusters of balanced cardinality. If we have to deal with ellipsoid shaped clusters, then the algorithm proposed by Gustafson and Kessel [149] would be helpful. This algorithm exploits adaptively modified Mahalanobis distance, defined in Section 2.2.1.2. The target function (3.72) has now the form

$$J_{\alpha,A_1,\dots,A_k}(U, M) = \sum_{j=1}^{k} \sum_{i=1}^{m} u_{ij}^{\alpha} \|\mathbf{x}_i - \boldsymbol{\mu}_j\|_{A_j}^2 \tag{3.80}$$

where $A_j \in \mathbb{R}^{n \times n}$ is a positive definite matrix defining the distances between the elements of the j-th cluster:

[24] K. Jajuga, L_1-norm based fuzzy clustering. *Fuzzy Sets and System* **39**, 43-50, 1991. See also P.J.F. Groenen, U. Kaymak and J. van Rosmalen: Fuzzy clustering with Minkowski distance functions, chapter 3 in [93].

$$\|\mathbf{x}_i - \boldsymbol{\mu}_j\|^2_{A_j} = (\mathbf{x}_i - \boldsymbol{\mu}_j)^{\mathrm{T}} A_j (\mathbf{x}_i - \boldsymbol{\mu}_j)$$

Matrix A_j is determined from the covariance matrix Σ_j, computed for objects of j-th cluster

$$A_j^{-1} = |\rho_j \Sigma_j|^{1/(r+1)} \Sigma_j^{-1} \tag{3.81}$$

where r is a user-defined parameter[25], ρ_j is the volume of the j-th cluster (usually $\rho = 1$), and Σ_j is the covariance matrix of the j-th cluster:

$$\Sigma_j = \frac{\sum_{i=1}^m u_{ij}^\alpha (\mathbf{x}_i - \boldsymbol{\mu}_j)(\mathbf{x}_i - \boldsymbol{\mu}_j)^{\mathrm{T}}}{\sum_{i=1}^m u_{ij}^\alpha} \tag{3.82}$$

and $|\Sigma_j|$ is the determinant of this matrix.

The algorithm consists of the following steps

1. Compute the prototypes of clusters $\boldsymbol{\mu}_j$ by applying the formula (3.57).
2. For each cluster compute the covariance matrix Σ_j from equation (3.82).
3. Calculate distances $d^2_{A_j}(\mathbf{x}_i, \boldsymbol{\mu}_j) = (\mathbf{x}_i - \boldsymbol{\mu}_j)^{\mathrm{T}} |\Sigma_j|^{1/n} \Sigma_j^{-1} (\mathbf{x}_i - \boldsymbol{\mu}_j)$
4. Update the assignment matrix U by assuming

$$u_{ij} = \left[\sum_{l=1}^k \left(\frac{d_{A_j}(\mathbf{x}_i, \boldsymbol{\mu}_j)}{d_{A_l}(\mathbf{x}_i, \boldsymbol{\mu}_l)} \right)^{\frac{2}{\alpha-1}} \right]^{-1}$$

This algorithm has been widely applied, in particular in digital image analysis[26] and in engineering tasks, [27]. However, if the coordinates of some of the observations constituting one group are strongly correlated then the covariance matrix suffers from singularities, which results in incorrectness of cluster assignments. A method immunising the algorithm against such situations was presented in [28].

Example 3.3.2 *Let us consider a synthetic data set consisting of three differently shaped clusters, presented in Figure 3.13. Samples representing the individual groups stem from two-dimensional normal distributions with parameters presented in Table 3.1.*

group	μ_X	μ_Y	σ_X	σ_Y
1	-0.5	0.5	0.35	0.15
2	0.7	0.7	0.05	0.25
3	0.5	-0.5	0.20	0.05

	$F_3(U)$	$H_3(U)$
FCM	0.8602	0.2865
GK	0.9133	0.1923

Table 3.1. Table to the left: Parameters of Gaussian distributions used to generate samples belonging to the three groups. Table to the right: Quality indicators of the partitions generated by the algorithms FCM and GK (Gustafson-Kessel)

[25] Its role is similar to that of the fuzziness exponent α. Usually we assume $r = n - 1$.

[26] R.N. Dave. Boundary detection through fuzzy clustering. In *IEEE International Conf. on Fuzzy Systems*, pp. 127–134, San Diego, USA, 1992

The table to the right presents the quality indicators for partitions generated by the FCM and GK (Gustafson-Kessel) algorithms. Both the partition coefficient and the partition entropy are better for the partition obtained from GK algorithm. Figure 3.13 shows silhouette diagrams of the membership functions generated by both algorithms. □

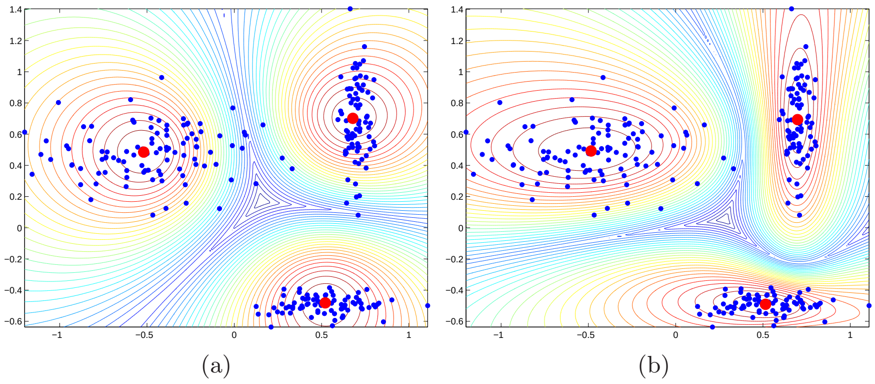


Fig. 3.13. Comparison of partitions generated by the (a) FCM algorithm and (b) GK algorithm for a synthetic data set consisting of three clusters with different shapes.

3.3.5.3 FCV algorithm: Fuzzy c-varietes

The FCM algorithm identifies spherical clusters by computing the centre and radius of each of them. The resultant spheres constitute *manifolds* describing data subsets. A natural research direction was to extend this idea to manifolds of other shapes. An extension to hyper-ellipsoids was presented in the preceding section. Bezdek introduced in [53] linear manifolds of dimension $r \in \{0, \ldots, s\}$, spanned over vectors $\{\mathbf{b}_1, \ldots, \mathbf{b}_r\}$ and passing through a given point $\mathbf{a} \in \mathbb{R}^s$. The subsequent equation describes the general form of such a manifold

$$\mathcal{V}^r(\mathbf{a}; \mathbf{b}_1, \ldots, \mathbf{b}_r) = \left\{ \mathbf{y} \in \mathbb{R}^s : \mathbf{y} = \mathbf{a} + \sum_{l=1}^{r} \xi_l \mathbf{b}_l, \xi_l \in \mathbb{R} \right\} \tag{3.83}$$

If $r = 0$ then $\mathcal{V}^0(\mathbf{a})$ is a point in s-dimensional Euclidean space; if $r = 1$ then $\mathcal{V}^1(\mathbf{a}; \mathbf{b})$ is a straight line in this space, and if $r = 2$ then $\mathcal{V}^2(\mathbf{a}; \mathbf{b}_1, \mathbf{b}_2)$ is a plane passing through the point $\mathbf{a} \in \mathbb{R}^s$.

Like in Section 3.3.5.2, one defines the distance $d_A(\mathbf{x}, \mathcal{V}^r)$ of the point $\mathbf{x} \in \mathbb{R}^s$ to the manifold $\mathcal{V}^r$:

$$d_A(\mathbf{x}, \mathcal{V}^r) = \min_{\mathbf{y} \in \mathbb{R}^s} d_A(\mathbf{x}, \mathbf{y}) = \left(\|\mathbf{x} - \mathbf{a}\|_A^2 - \sum_{l=1}^{r} \left((\mathbf{x} - \mathbf{a})^{\mathsf{T}} A \mathbf{b}_l \right)^2 \right)^{1/2} \tag{3.84}$$

If we have now k manifolds $\mathcal{V}_1^r, \dots, \mathcal{V}_k^r$ described by vectors $\mathfrak{a} = (\mathbf{a}_1, \dots, \mathbf{a}_k)$, $\mathfrak{b}_j = (\mathbf{b}_{1j}, \dots, \mathbf{b}_{kj})$, $j = 1, \dots, r$, i.e.

$$\mathcal{V}_j^r = \left\{ \mathbf{y} \in \mathbb{R}^s \colon \mathbf{a}_j + \sum_{l=1}^{r} \xi_l \mathbf{b}_{jl},\ \xi_l \in \mathbb{R} \right\},\ j = 1, \dots, k$$

then the generalised target function (3.72) has the form

$$J_\alpha(U, \mathfrak{a}; \mathfrak{b}_1, \dots, \mathfrak{b}_r) = \sum_{j=1}^{k} \sum_{i=1}^{m} u_{ij}^m D_{ij}^2 \tag{3.85}$$

where $D_{ij} = d_A(\mathbf{x}_i, \mathcal{V}_j^r)$.

Once again the minimisation of this indicator is achieved via a variant of the alternating algorithm from Section 3.3.2. Vectors $\mathbf{a}_j$ and $\mathbf{b}_{jl}$, $l = 1, \dots, r$, spanning the j-th manifold are determined from the equation [27]

$$\mathbf{a}_j = \frac{\sum_{i=1}^m u_{ij}^\alpha \mathbf{x}_i}{\sum_{i=1}^m u_{ij}^\alpha}, \quad \mathbf{b}_{jl} = A^{-1/2} s_{jl} \tag{3.86}$$

where s_{jl} is the l-th eigenvector of the scatter matrix

$$S_j = A^{1/2} \left(\sum_{i=1}^{m} u_{ij}^\alpha (\mathbf{x}_i - \mathbf{a}_j)(\mathbf{x}_i - \mathbf{a}_j)^{\mathsf{T}} \right) A^{1/2}$$

corresponding to the l-th maximal eigenvalue[28] of this matrix. Membership degree of i-th object in j-th manifold is determined using an analogue of the equation (3.60), i.e.

$$u_{ij} = \frac{D_{ij}^{\frac{2}{1-\alpha}}}{\sum_{l=1}^k D_{il}^{\frac{2}{1-\alpha}}} \tag{3.87}$$

The algorithm presented here is very sensitive to initialisation, particularly for bigger values of the parameter k. This topic has been investigated more deeply in e.g. [217] and [373].

The reader may pay attention to the fact that the FCV algorithm can be viewed as a precursor of approaches like k-planes algorithm presented in [58], projective clustering [107], or manifold learning [304]. It is also worth mentioning that a simple clustering algorithm for straight-line-segment shaped clusters was presented in [126].

[27] See J.C. Bezdek, C. Coray, R. Gunderson and J. Watson. Detection and characterization of cluster substructure: I. Linear structure: Fuzzy c-lines. *SIAM J. Appl. Math.* **40**(2), 339-357, 1981; Part II Fuzzy c-varieties and convex combinations thereof, *SIAM J. Appl. Math.* **40**(2), 358-372, 1981.

[28] Note that S_j is a symmetric matrix of dimension $n \times n$. Hence it has n (not necessarily distinct) real-valued eigenvalues. See Section B.3.

3.3.5.4 FCS algorithm: Fuzzy c-shells

Still another generalisation of the basic FCM algorithm was proposed by Dave [88], [90]. His algorithm was devoted to detection of spherical and elliptical edges in digital images, as an alternative approach to Circle Hough Transform for circle detection. Though Hough transform, [144], is a useful edge detection technique, it has the drawback of high memory and time complexity.

The FCS algorithm seeks a fuzzy partition into k circular groups. Each group is determined by the circle centre $\mathfrak{a} = \{\mathbf{a}_1, \dots, \mathbf{a}_k\}$ and radius $\mathbf{r} = \{r_1, \dots, r_k\}$. The algorithm strives to minimize the indicator

$$J_\alpha(U, \mathfrak{a}, \mathbf{r}) = \sum_{j=1}^{k} \sum_{i=1}^{m} u_{ik}^\alpha \Big(\|\mathbf{x}_i - \mathbf{a}_j\|_{A_j} - r_j \Big)^2 \tag{3.88}$$

We assume here that $\mathbf{a}_j \in \mathbb{R}^s$, $r_j > 0$, $j = 1, \dots, k$.

Under these circumstances, the FCS algorithm consists of the following steps:

(i) Fix the algorithm parameters α, k and the precision ϵ.
(ii) Initialise the partition matrix $U^{(t)}$
(iii) Determine circle centres $\mathbf{a}_j$ and radiuses r_j by solving the equation system

$$\sum_{i=1}^{m} u_{ij}^\alpha \left(1 - \frac{r_j}{\|\mathbf{x}_i - \mathbf{a}_j\|_{A_j}} \right) (\mathbf{x}_i - \mathbf{a}_j) = 0 \tag{3.89}$$

and

$$\sum_{i=1}^{m} u_{ij}^\alpha \big(\|\mathbf{x}_i - \mathbf{a}_j\|_{A_j} - r_j \big) = 0 \tag{3.90}$$

(iv) Determine the new membership degrees u_{ij} by applying equation (3.87) with $D_{ij} = \big(\|\mathbf{x}_i - \mathbf{a}_j\|_{A_j} - r_j \big)$.
(v) if $\|U_{old} - U_{new}\| < \epsilon$ then STOP, else return to step (iii).

If $A_j = \mathbb{I}$, $j = 1, \dots, k$, then the above algorithm detects clusters, in which objects are located on or in close vicinity to circles, representing individual groups. By assigning each group a unique matrix A_j one obtains a generalised (adaptive) FCS algorithm. In such a case, when new values $\mathbf{a}_j$, r_j are determined, one needs to update the matrix A_j using e.g. the equation (3.81).

A careful convergence analysis performed for the FCS algorithm in [52] suggests that a single iteration of the Newton method should be used to determine subsequent approximations of the vector $\mathbf{a}_j$ and the radius r_j. On the other hand, Dave finds in [90] that already three iterations of the Newton method are sufficient to reach precise results.

One can approach manifolds of other shapes in a way analogous to the one presented above[29].

[29] See e.g. F. Klawonn, R. Kruse, and H. Timm. Fuzzy shell cluster analysis. *Courses and Lectures – International Centre for Mechanical Sciences*,

3.3.5.5 SFCM: spherical FCM algorithm

If both $\mathbf{x}_i$ and $\boldsymbol{\mu}_j$ are unit vectors, we can recall the observation (3.11) and define the target function as follows

$$J_s(U,M) = \sum_{i=1}^{m}\sum_{j=1}^{k} u_{ij}^{\alpha}\Big(1 - \sum_{l=1}^{n} x_{il}\mu_{jl}\Big) \tag{3.91}$$

We seek such a pair (U^*, M^*), for which the above function attains the minimum. This time, we require not only that $U \in \mathcal{U}_{fk}$ but also that $\|\boldsymbol{\mu}_j\| = 1$. The latter condition means that

$$\sum_{l=1}^{n} \mu_{jl}^2 = 1,\ j = 1,\ldots,k \tag{3.92}$$

Here, the Lagrange function has the form

$$L(J_s,\lambda,\Lambda) = \sum_{i=1}^{m}\sum_{j=1}^{k} u_{ij}^{\alpha}\Big(1-\sum_{l=1}^{n} x_{il}\Big) - \sum_{i=1}^{m}\lambda_i\Big(\sum_{j=1}^{k} u_{ij} - 1\Big) - \sum_{j=1}^{k}\Lambda_j\Big(\sum_{l=1}^{n}\mu_{jl}^2 - 1\Big) \tag{3.93}$$

By applying a procedure analogous to the one from Appendix A we can state that the entries of the matrix U are of the form

$$u_{ij} = \frac{D^{\frac{1}{1-\alpha}}(\mathbf{x}_i,\boldsymbol{\mu}_j)}{\sum_{l=1}^{k} D^{\frac{1}{1-\alpha}}(\mathbf{x}_i,\boldsymbol{\mu}_l)} = \frac{\Big(1-\sum_{t=1}^{n} x_{it}\mu_{jt}\Big)^{\frac{1}{1-\alpha}}}{\sum_{l=1}^{k}\Big(1-\sum_{t=1}^{n} x_{it}\mu_{lt}\Big)^{\frac{1}{1-\alpha}}} \tag{3.94}$$

The above formula holds only if $D(\mathbf{x}_i,\boldsymbol{\mu}_j) = (1 - \mathbf{x}_i^{\mathsf{T}}\boldsymbol{\mu}_j) \neq 0$. Otherwise, if the constraint does not hold, one applies a variant of the equation (3.58).

By solving the equation system

$$\begin{cases} \dfrac{\partial}{\partial\boldsymbol{\mu}_{jl}} L(J_s,\lambda,\Lambda) = 0 \\ \dfrac{\partial}{\partial\Lambda_j,} L(J_\alpha,\lambda,\Lambda) = 0 \end{cases} \quad j = 1,\ldots,k, \quad l = 1,\ldots,n \tag{3.95}$$

one determines the prototype components, see [254], as

$$\boldsymbol{\mu}_{jl} = \sum_{i=1}^{m} u_{ij}^{\alpha} x_{il}\sqrt{\sum_{r=1}^{n}\Big(\sum_{t=1}^{m} u_{jt}^{\alpha} x_{tr}\Big)^{-2}} \tag{3.96}$$

Springer 1997, pp. 105-120; `http://public.fh-wolfenbuettel.de/~klawonn/Papers/klawonnetaludine97.pdf`.

3.3.5.6 Kernel-based variants of the FCM algorithm

Like classical k-means algorithm, also FCM algorithm can be kernelised. Two variants of the kernel-based fuzzy c-means algorithm are distinguished. The KFCM-X variant constructs the class prototypes in the original feature space. The other variant, called KFCM-$\mathcal{F}$, assumes that the prototypes are defined in the kernel feature space, in analogy to the KHCM algorithm from Section 3.1.5.5.

3.3.5.6.1 KFCM-X algorithm

The algorithm seeks the minimum of the target function defined as

$$J_\alpha^K(U,M) = \sum_{i=1}^{m}\sum_{j=1}^{k} u_{ij}^\alpha \|\Phi(\mathbf{x}_i) - \Phi(\boldsymbol{\mu}_j)\|^2 \tag{3.97}$$

We know already that

$$\|\Phi(\mathbf{x}_i) - \Phi(\boldsymbol{\mu}_j)\|^2 = \mathsf{K}(\mathbf{x}_i,\mathbf{x}_i) - 2\mathsf{K}(\mathbf{x}_i,\boldsymbol{\mu}_j) + \mathsf{K}(\boldsymbol{\mu}_j,\boldsymbol{\mu}_j) \tag{3.98}$$

where $\mathsf{K}(\mathbf{x},\mathbf{y}) = \Phi^{\mathsf{T}}(\mathbf{x})\Phi(\mathbf{y})$ is a kernel function. The most frequently applied function is in practice the Gaussian kernel, i.e. $\mathsf{K}(\mathbf{x},\mathbf{y}) = \exp\big(\|\mathbf{x}-\mathbf{y}\|^2/\sigma^2\big)$, where $\sigma > 0$ is a parameter. In such a case $\mathsf{K}(\mathbf{x},\mathbf{x}) = 1$, hence[30]

$$\|\Phi(\mathbf{x}_i) - \Phi(\boldsymbol{\mu}_j)\|^2 = 2\big(1 - \mathsf{K}(\mathbf{x}_i,\boldsymbol{\mu}_j)\big)$$

which means that the quality index (3.97) is of the form

$$J_\alpha^K(U,M) = 2\sum_{i=1}^{m}\sum_{j=1}^{k} u_{ij}^\alpha (1 - \mathsf{K}(\mathbf{x}_i,\boldsymbol{\mu}_j) \tag{3.99}$$

The resulting algorithm resembles the original FCM algorithm where one alternates between updating the membership degrees u_{ij} and the prototype coordinates according to the equations below[31], see e.g. [147], [385]:

$$u_{ij} = \frac{\big(1 - \mathsf{K}(\mathbf{x}_i,\boldsymbol{\mu}_j)\big)^{\frac{1}{1-\alpha}}}{\sum_{l=1}^{k}\big(1 - \mathsf{K}(\mathbf{x}_i,\boldsymbol{\mu}_l)\big)^{\frac{1}{1-\alpha}}} \tag{3.100}$$

$$\boldsymbol{\mu}_j = \frac{\sum_{i=1}^{m} u_{ij}^\alpha \mathsf{K}(\mathbf{x}_i,\boldsymbol{\mu}_j)\mathbf{x}_i}{\sum_{i=1}^{m} u_{ij}^\alpha \mathsf{K}(\mathbf{x}_i,\boldsymbol{\mu}_j)} \tag{3.101}$$

If K is a Gaussian kernel and $\sigma \to \infty$, then $\mathsf{K}(\mathbf{x}_i,\boldsymbol{\mu}_j) \to 1 - \|\mathbf{x}_i - \boldsymbol{\mu}_j)\|^2/\sigma^2$ and the algorithm resembles the performance of the classical FCM algorithm.

[30] In fact the formulas below remain valid for any kernel function matching the condition $K(\mathbf{x},\mathbf{x}) = 1$.

[31] If $\mathsf{K}(\mathbf{x}_i,\boldsymbol{\mu}_l) = 1$ for some index l then, upon determining u_{il}, we proceed analogously as when applying equation (3.58)

The procedure was applied in [385] to cluster incomplete data. Under such scenario the algorithm consists of the following steps

(a) Initialise prototypes $\boldsymbol{\mu}_j$, $j = 1, \dots, k$.
(b) Substitute $x_{ib} = 0$, if b-th component of i-th observation is not known.
(c) As long as prototypes are not stable, execute:
  (c1) update membership degrees according to equation (3.100),
  (c2) update prototypes according to equation (3.101),
  (c3) reconstruct values of unknown features by applying a variant of equation (3.67):

$$x_{ib} = \frac{\sum_{j=1}^{k} u_{ij}^{\alpha} K(\mathbf{x}_i, \boldsymbol{\mu}_j)\mu_{jb}}{\sum_{j=1}^{k} u_{ij}^{\alpha} K(\mathbf{x}_i, \boldsymbol{\mu}_j)}$$

3.3.5.6.2 KFCM-$\mathcal{F}$ algorithm

In this case the target function is of the form

$$J_{\alpha}^{\mathcal{F}}(U) = \sum_{i=1}^{m}\sum_{j=1}^{k} u_{ij}^{\alpha} \|\Phi(\mathbf{x}_i) - \boldsymbol{\mu}_j^{\Phi}\|^2 \tag{3.102}$$

Following the guidelines of Section 3.1.5.5 and [147] and the references cited therein, the membership degrees u_{ij} are determined from the equation

$$u_{ij} = \frac{\|\Phi(\mathbf{x}_i) - \boldsymbol{\mu}_j^{\Phi}\|^{\frac{2}{1-\alpha}}}{\sum_{l=1}^{k} \|\Phi(\mathbf{x}_i) - \boldsymbol{\mu}_l^{\Phi}\|^{\frac{2}{1-\alpha}}} \tag{3.103}$$

Because of

$$\|\Phi(\mathbf{x}_i) - \boldsymbol{\mu}_j^{\Phi}\|^2 = \Phi(\mathbf{x}_i)^{\mathsf{T}}\Phi(\mathbf{x}_i) - 2\Phi(\mathbf{x}_i)\boldsymbol{\mu}_j^{\Phi} + (\boldsymbol{\mu}_j^{\Phi})^{\mathsf{T}}\boldsymbol{\mu}_j$$

and

$$\boldsymbol{\mu}_j^{\Phi} = \frac{\sum_{i=1}^{m} u_{ij}^{\alpha}\Phi(\mathbf{x}_i)}{\sum_{i=1}^{m} u_{ij}^{\alpha}} \tag{3.104}$$

we obtain

$$\|\Phi(\mathbf{x}_i) - \boldsymbol{\mu}_j^{\Phi}\|^2 = \mathsf{K}(\mathbf{x}_i, \mathbf{x}_i) - 2\frac{\sum_{l=1}^{m} u_{lj}^{\alpha}\mathsf{K}(\mathbf{x}_i, \mathbf{x}_l)}{\sum_{l=1}^{m} u_{lj}^{\alpha}} + \frac{\sum_{l=1}^{m}\sum_{t=1}^{m} u_{lj}^{\alpha}u_{tj}^{\alpha}\mathsf{K}(\mathbf{x}_i, \mathbf{x}_t)}{\left(\sum_{l=1}^{m} u_{lj}^{\alpha}\right)^2} \tag{3.105}$$

Like in the case of the kernel-based variant of the k-means algorithm, there exists also the possibility to construct the matrix K with elements $k_{ij} = \mathsf{K}(\mathbf{x}_i, \mathbf{x}_j)$ in advance. Therefore the above-mentioned equation can be simplified to

$$\|\Phi(\mathbf{x}_i) - \boldsymbol{\mu}_j\|^2 = k_{ii} - 2\frac{\sum_{l=1}^{m} u_{lj}^{\alpha}k_{il}}{\sum_{l=1}^{m} u_{lj}^{\alpha}} + \frac{\sum_{l=1}^{m}\sum_{t=1}^{m} u_{lj}^{\alpha}u_{tj}^{\alpha}k_{lt}}{\left(\sum\limits_{l=1}^{m} u_{lj}^{\alpha}\right)^2} \tag{3.106}$$

or in vector form (consult equation (3.27) from Section 3.1.5.5):

$$\|\Phi(\mathbf{x}_i) - \boldsymbol{\mu}_j\|^2 = k_{ii} - 2(\widetilde{\mathbf{u}}_j^\mathsf{T} K)_i + \widetilde{\mathbf{u}}_j^\mathsf{T} K \widetilde{\mathbf{u}}_j \tag{3.107}$$

where $\mathbf{u}_j$ is the j-th column of matrix U, and $\widetilde{\mathbf{u}}_j = \mathbf{u}_j^\alpha / \|\mathbf{u}_j^\alpha\|_1$.

So, the kernel-based analogue of the FCM algorithm, similar to the algorithm 3.6 from page 88 runs as follows: squares of distances of objects to prototypes, computed from the above formula, are substituted into the equation (3.103) and the computations are repeated until stability is reached of all values of u_{ij}.

Zhou and Gan proposed in [394] an interesting idea of determining group prototypes. $\boldsymbol{\mu}_j^\Phi$ in (kernel) feature space has the counterimage $\mathbf{m}_j$ in the original feature space in which the expression $\|\Phi(\mathbf{m}_j) - \boldsymbol{\mu}_j^\Phi\|$ arrives at its minimum. By substituting (3.104) into this equation and setting to zero the gradient (with respect to $\mathbf{m}_j$) of an equation modified in this way one gets the formula

$$\mathbf{m}_j \sum_{i=1}^m u_{ij}^\alpha \mathsf{K}(\mathbf{x}_i, \mathbf{m}_j) = \sum_{i=1}^m u_{ij}^\alpha \mathsf{K}(\mathbf{x}_i, \mathbf{m}_j)\mathbf{x}_i \tag{3.108}$$

If K is a Gaussian kernel, we get the expression

$$\mathbf{m}_j = \frac{\sum_{i=1}^m u_{ij}^\alpha \mathsf{K}(\mathbf{x}_i, \mathbf{m}_j)\mathbf{x}_i}{\sum_{i=1}^m u_{ij}^\alpha \mathsf{K}(\mathbf{x}_i, \mathbf{m}_j)} \tag{3.109}$$

After initiating the vector $\mathbf{m}_j \in \mathbb{R}^n$, the computations are repeated following the above equation until the solution is stable.

The paper [171] proposes modifications of the kernel-based fuzzy algorithms, meant to adapt them to processing big data sets.

Application of the algorithms described above should be cautious. An exhaustive study [147] compares the algorithms FCM, GK (Gustafson and Kessel) with KFCM-X and KFCM-$\mathfrak{F}$ for a large set of test datasets. The authors of that paper draw quite an important conclusion:

> The kernel-based clustering algorithms – especially KFCM-$\mathfrak{F}$ – can cluster specific non-spherical clusters such as the ring cluster quite well outperforming FCM and GK for the same number of clusters; however, overall the performance of the kernel-based methods is not very impressive due to similar or only slight increases in clustering classification rates compared to FCM. From the perspective of the reconstruction error, KFCM-$\mathfrak{F}$ often performed similar to that of FCM and KFCM-X. The major disadvantages of kernel-based algorithms are:
> – selection of the kernel function and
> – optimisation of kernel parameters.
> (...) Generally there is no statistically significant difference between the kernel-based algorithms and the FCM and GK algorithms except in a few instances.

3.3.5.7 PCM: possibilistic clustering algorithm

Much attention has been paid to methods strengthening the resistance of FCM algorithm to noise and outliers. One of such approaches was presented in [89], where the set of k groups was extended by one additional cluster that is intended to contain all non-typical data. This approach redefines the quality indicator as follows:

$$J_\alpha(U,V) = \sum_{i=1}^{m}\sum_{j=1}^{k} u_{ij}^{\alpha} d^2(\mathbf{x}_i, v_j) + \sum_{i=1}^{m} u_{i,k+1}^{\alpha}\delta^2 \tag{3.110}$$

where δ represents the distance of an object from the "noisy" cluster. Note that this distance is identical for all objects.

Wu and Yang, on the other hand, introduced in [366] the target function of the form

$$J_\alpha(U,M;\beta) = \sum_{j=1}^{k}\sum_{i=1}^{m} u_{ij}^{\alpha}\Big(1 - \exp(-\beta\|\mathbf{x}-\mathbf{y})\|^2\Big) \tag{3.111}$$

where

$$\beta = \frac{m}{\sum_{i=1}^{m}\|\mathbf{x}_i - \overline{\mathbf{x}}\|^2}, \quad \overline{\mathbf{x}} = \frac{1}{m}\sum_{i=1}^{m}\mathbf{x}_i$$

The authors claim that not only the resulting algorithm is resistant to the presence of noise and outliers but also it tolerates groups with diverse cardinalities.

Another interesting proposal is to resign from the requirement that the total membership degree of an objects to all k clusters be equal 1, $\sum_{j=1}^{k} u_{ij} = 1$, as this requirement forces each outlier to be assigned to one or more clusters.

We say that the matrix U represents a possibilistic partition of an object set if the following three conditions are met:

$$(a)\ 0 \le u_{ij} \le 1,\ i = 1,\dots,m, j = 1,\dots,c$$

$$(b)\ \sum_{i=1}^{m} u_{ij} > 0,\ j = 1,\dots,k$$

$$(c)\ \sum_{j=1}^{c} u_{ij} > 0,\ i = 1,\dots,m$$

In order to obtain such a non-trivial partition, Krishnapuram and Keller, [216] introduced a target function of the form:

$$J_P(U,M) = \sum_{j=1}^{k}\sum_{i=1}^{m} u_{ij}^{\alpha}\|\mathbf{x}_i - \boldsymbol{\mu}_j\|^2 + \sum_{j=1}^{k}\eta_j\sum_{i=1}^{m}(1-u_{ij})^{\alpha} \tag{3.112}$$

where $\eta_j > 0$, $j = 1, \dots, k$. The role of the second summand is to enforce a large value of the element u_{ij}. Elements u_{ij} are now treated as degrees of typicality and each column of the matrix U represents the so called distribution of possibilities over the set of objects.

Parameter η_j controls the "typicality" of objects: it indicates the distance from the prototype below which it is permissible to assign high membership values u_{ij}. The value u_{ij} is obtained from the formula

$$u_{ij} = \left(1 + \frac{\|\mathbf{x}_i - \boldsymbol{\mu}_j\|^2}{\eta_j}\right)^{\frac{1}{\alpha-1}} \tag{3.113}$$

The membership degree of the object $\mathbf{x}_i$ in the class j in the standard FCM depends on the distance from each prototype . But this is different in the PCM algorithm. Now, u_{ij} depends on the distance from the prototype $\boldsymbol{\mu}_j$ alone, with this distance being additionally modified by the parameter η_j. If $\eta_j = \|\mathbf{x}_i - \boldsymbol{\mu}_j\|^2$ then $u_{ij} = 1/2$. Hence η_j points at the distance from the prototype $\boldsymbol{\mu}_j$, for which u_{ij} reaches the value 1/2. When $u_{ij} > 1/2$, then $\mathbf{x}_i$ is treated as a typical representative of the cluster C_j. Krishnapuram and Keller propose in [216] to compute η_j from the equation

$$\eta_j = \frac{\sum_{i=1}^{m} u_{ij}^{\alpha}\|\mathbf{x}_i - \boldsymbol{\mu}_j\|^2}{\sum_{i=1}^{m} u_{ij}^{\alpha}} \tag{3.114}$$

This value can be updated in each iteration or fixed during initialisation. The first variant may be the source of instability of the algorithm, therefore the second variant is suggested in [216] with u_{ij} being substituted with values returned by the FCM algorithm. After initialising the matrix U and determining the value η_j, the prototype coordinates are computed as prescribed by equation (3.57) and new values of assignments are computed from equation (3.113). These last two steps are repeated till the values of the matrix U are stable.

The PCM algorithm has two important drawbacks: it is sensitive to initialisation and the function (3.112) reaches minimum when the coordinates of all prototypes are identical[32], which is a consequence of the fact that the degree u_{ij} depends only on the distance of the i-th object from the j-th cluster. Authors cited in the footnote proposed to enrich the function (3.112) with a summand preventing collapsing of prototypes:

$$J_{RP}(U,M) = \sum_{j=1}^{k}\sum_{i=1}^{m} u_{ij}^{\alpha}\|\mathbf{x}_i - \boldsymbol{\mu}_j\|^2 + \sum_{j=1}^{k}\eta_j\sum_{i=1}^{m}(1-u_{ij})^{\alpha} + \sum_{i=1}^{k}\gamma_i\sum_{j=1, j\neq i}^{k}\|\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\|^{-2} \tag{3.115}$$

In this case the coordinates of the prototypes are updated as follows:

[32] This problem has been analysed in the paper by H. Timm, C. Borgelt, C. Doring, and R. Kruse, "An extension to possibilistic fuzzy cluster analysis", *Fuzzy Sets Syst.*, **147**(1), 2004, pp. 3–16.

$$\boldsymbol{\mu}_j = \frac{\sum_{i=1}^m u_{ij}^\alpha \mathbf{x}_i - \gamma_j \sum_{l=1,l\neq j}^k \boldsymbol{\mu}_l r_{jl}}{\sum_{i=1}^m u_{ij}^\alpha - \gamma_j \sum_{l=1,l\neq j}^k r_{jl}} \tag{3.116}$$

where $r_{jl} = \|\boldsymbol{\mu}_j - \boldsymbol{\mu}_l\|^{-4}$ means the repulsion force between respective prototypes. Values u_{ij} are determined according to equation (3.113). If the repulsion force exceeds the attraction force, i.e. $\sum_{i=1}^m u_{ij} < \gamma_j \sum_{l\neq j} r_{il}$ of one (or more) prototype $\boldsymbol{\mu}_j$, then such a prototype should be initialised anew.

Pal *et al.* proposed in [277] to combine the FCM algorithm with the PCM algorithm in such a way that at the same time the membership degrees u_{ij} and the typicality degrees t_{ij} are computed and the target function is formalised as follows:

$$J_{RP}(U,M) = \sum_{j=1}^k \sum_{i=1}^m (a u_{ij}^\alpha + b t_{ij}^\beta \|\mathbf{x}_i - \boldsymbol{\mu}_j\|^2 + \sum_{j=1}^k \eta_j \sum_{i=1}^m (1 - t_{ij})^\beta \tag{3.117}$$

where the parameter $a > 0$ characterises the importance of the membership degrees, and the parameter $b > 0$ stresses the importance of typicality degrees. If $b = 0$ then the algorithm behaves like FCM. It is required that the exponents respect the constraint $\alpha, \beta > 0$. It is recommended that they be equal $\alpha = \beta = 2.0$.

More information on possibilistic clustering algorithms can be found in [277] and in the bibliography cited therein[33]. Kernel based variants of such algorithms are discussed in [125].

3.3.5.8 Relational variant of the FCM algorithm

Let us consider now the task of clustering objects in a situation where their description in terms of a feature vector is not available and, instead, we can access only their dissimilarity matrix R. Its elements r_{ij} are interpreted as dissimilarity degrees of the i-th object from the j-th object. We assume that r is a symmetric matrix with non-negative elements, and additionally $r_{ii} = 0$ for all $i = 1, \ldots, m$.

This problem was investigated, among others, by Roubens [295] and Windham [362]. The algorithms they propose suffer, however, from a strong dependence on initialisation, which influences the stability of the algorithms and the quality of the generated clusters which is usually poor.

Hathaway, Davenport and Bezdek [170] developed a more satisfactory algorithm, but at the expense of additional constraints on the matrix R. They require that the dissimilarity matrix be Euclidean, meaning that in some n'-dimensional space there exists such a set of points $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ that $r_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|^2$. Let also U be an (arbitrarily initialised) matrix of dimension $m \times k$, representing the

[33] We recommend also the paper X. Wu, B. Wu, J. Sun and H. Fu, "Unsupervised possibilistic fuzzy clustering", *J. of Information & Computational Science*, **7**(5), 2010, pp. 1075-1080.

group membership of objects, i.e. a matrix fulfilling the conditions (3.52). Under these assumptions the prototype ("centre") of the j-th cluster is determined from the equation

$$\boldsymbol{\mu}_j = \frac{(u_{1j}^\alpha, \dots, u_{mj}^\alpha)^{\mathtt{T}}}{\sum_{i=1}^m u_{ij}^\alpha}, \ j = 1, \dots, k \tag{3.118}$$

and the distance [34] d_{ij} between the i-th object and the j-th prototype is calculated from the equation

$$d_{ij} = (R\boldsymbol{\mu}_j)_i - \frac{1}{2}\boldsymbol{\mu}_j^{\mathtt{T}} R \boldsymbol{\mu}_j, i = 1, \dots, m, j = 1, \dots, k \tag{3.119}$$

where the symbol $(\mathbf{v})_i$ denotes the i-th element of the vector $\mathbf{v}$.

New values u_{ij} of the membership of i-th object in the j-th group are obtained from equation (3.58), i.e.

$$u_{ij} = \begin{cases} \left[\sum_{l=1}^{k} \left(\frac{d_{ij}}{d_{il}} \right)^{\frac{1}{\alpha-1}} \right]^{-1} & \text{if } Z_i = \emptyset \\ 1 & \text{if } j \in Z_i \neq \emptyset \\ 0 & \text{if } j \notin Z_i \neq \emptyset \end{cases} \tag{3.120}$$

where $Z_i = \{j \colon 1 \le j \le k, d_{ij} = 0\}$, as before.

If $r_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|^2$ then the partition returned by the above algorithm is identical with the partition returned by the classical FCM algorithm. If, however, R is not a Euclidean dissimilarity relation then the values d_{ij}^2, obtained from equation (3.119) may be negative. Non-Euclidean relation R can be transformed to a Euclidean one using a simple mapping, [168]

$$r_{ij}^\beta = \begin{cases} r_{ij} + \beta & \text{if } i \neq j \\ 0 & \text{if } i = j \end{cases}$$

where $\beta \ge \beta_0$, and β_0 is some positive number. In section 3 of the paper [168] an adaptive procedure was presented, enabling to estimate the appropriate value of the parameter β.

Another, simpler solution of this problem was proposed in [84]. The respective authors suggested that information about relations between the objects allows to describe the i-th object with the vector $\mathbf{x}_i = (r_{i,1}, \dots, r_{i,m})^{\mathtt{T}}$ expressing the dissimilarity of this object with respect to the other ones. Also, the prototype $\boldsymbol{\mu}_j$ of the j-th class can be represented in the same way. Hence, one can define the gap between the dissimilarity of the i-th object with respect to the other objects and the profile of the j-th class, that is

$$\delta^2(\mathbf{x}_i, \boldsymbol{\mu}_j) = \sum_{l=1}^{m} (r_{il} - \mu_{jl})^2 \tag{3.121}$$

[34] Formally it is the squared distance.

and then try to minimise the indicator

$$J_\alpha(U,C) = \sum_{i=1}^{m}\sum_{j=1}^{k} u_{ij}^\alpha \delta^2(\mathbf{x}_i, \boldsymbol{\mu}_j) \tag{3.122}$$

where, as previously, elements u_{ij} of the assignment matrix U fulfil the conditions $u_{ij} \in [0,1]$, $\sum_{j=1}^{k} u_{ij} = 1$, and $\sum_{i=1}^{m} u_{ij} > 0$.

Equation (3.122) has the same shape as equation (3.53), which means that equations (3.57) and (3.58) can be applied to determine the prototypes and the assignment (allocation) matrix. Under the assumed representation of prototypes, the first of these equations will have the form

$$\boldsymbol{\mu}_{j,l} = \frac{\sum_{s=1}^{m} u_{lj}^\alpha \cdot r_{ls}}{\sum_{s=1}^{m} u_{sj}}, j = 1,\ldots,k, l = 1,\ldots,m \tag{3.123}$$

Iterative repetitions of computations, prescribed by equations (3.123) and (3.58) constitute the ARCA algorithm (*Any Relation Clustering Algorithm*). Like FCM, it requires initialisation of the matrix U. Authors of the paper [84] assumed, following Windham in [362], that $u_{i,1} = 0.5(1 + 1/k)$ for the first m/k objects, $u_{i,2} = 0.5(1 + 1/k)$ for the next m/k objects, ..., $u_{i,k} = 0.5(1 + 1/k)$ for the remaining m/k objects, and membership of all objects in other classes is equal $u_{ij} = 0.5/k$.

More information on the relational variant of the FCM algorithm can be found in chapter 3 of the book [53]. Its applications in data mining, in particular in personalisation of search results, are discussed, among others, in [215], [266].

3.4 Affinity propagation

We now deal with another example of a relational algorithm returning a set of prototypes[35]. Input data consist of similarities between pairs of objects, s_{ij}. Authors of [135] assume that the number s_{ij}, $i \neq j$ indicates how well the point j is suitable to represent the point i. In the simplest case $s_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|^2$, though one can apply other similarity measures (see examples in the paper [135]). The numbers s_{ii} describe so called preferences, the bigger is s_{ii}, the greater is the chance that the object i will become a prototype. s_{ii} can be equal e.g. to the median of the set of values s_{ij}, $i \neq j$ over all j (resulting in a larger number of clusters) or the minimal value from this set (whereby a smaller number of clusters is induced). Neither symmetry nor metricity of the similarities is required (i.e. the condition $s_{ik} < s_{ij} + s_{jk}$ is not imposed).

The essence of the algorithm is to maximise the function

$$E(\mathbf{c}) = -\sum_{i=1}^{m} s_{i,c_i} \tag{3.124}$$

[35] Its authors call those prototypes *exemplars* (of data groups) [135].

where $\mathbf{c} = (c_1, \ldots, c_m)$ is a set of labels. Label c_i points at the representative of the object i. A task formulated in this way is $\mathcal{NP}$-hard, therefore it is formulated as maximisation of the *pure* similarity of the form

$$E(\mathbf{c}) = -\sum_{i=1}^{m} s_{i,c_i} + \sum_{j=1}^{m} \delta_j(\mathbf{c}) \tag{3.125}$$

where

$$\delta_j(\mathbf{c}) = \begin{cases} -\infty \text{ if } c_j \neq j \text{ and } \exists i\colon c_i = j \\ \quad 0 \text{ otherwise} \end{cases} \tag{3.126}$$

is the penalty imposed in case object i chooses object $j = c_i$ as its prototype and object j does not consider itself as its own prototype, that is $c_j \neq j$. It is an essential condition that a prototype has to fulfil: not only several objects should choose j as their representative, but also j must choose itself as its own prototype, i.e. the following condition must hold: $c_j = j$.

The problem (3.125) is solved in a process of message passing, hence its name *affinity propagation*. Message r_{ij} sent by the object i to the object j reflects *responsibility* of being the prototype for object i. *Availability*, a_{ij}, is a message sent by the object j to the object i informing about the readiness to take over the task of being the prototype for the object i.

If we disregard technicalities, related to the message exchange[36], then the algorithm has the form shown as pseudo-code 3.10. Most important operations therein are the updates of responsibility and availability, described by equations (3.127) and (3.129). In the first iteration, the new values of responsibility are equal

$$r'_{ij} = \begin{cases} s_{ij} - s_{i,j_1(i)} \text{ if } j \neq j_1(i) \\ s_{ij} - s_{i,j_2(i)} \text{ otherwise} \end{cases}, \quad i, j = 1, \ldots, m$$

where $j_1(i)$ is the id of the object most similar to object i, while $j_2(i)$ is the id of the object second most similar to object i.

A "smoothing" operation, described by equations (3.128) and (3.130), was introduced in order to avoid numerical oscillations of the values of both messages.

The algorithm terminates when either a predefined number of **while** loop iterations was performed or when the object assignments to the prototypes do not change over t consecutive iterations (in [135] $t = 10$ was assumed). Values c_i, defined by the equation (3.131), indicate the prototypes (if $c_i = i$) or membership of the object i in the class represented by the appropriate prototype (if $c_i \neq i$).

Please note that the number of classes does not need to be specified in advance, contrary to the algorithms discussed previously. The affinity propagation algorithm determines it automatically.

[36] Interested reader is recommended to get acquainted with the PhD Thesis [113].

Algorithm 3.10 Affinity propagation, [135]

```
Require: Similarity matrix $S = [s_ij]_{m\times m}$.
Ensure: Set of prototypes together with the objects represented by them.
1: $a_{ij} = 0$ for $i, j = 1, \dots, m$
2: while (not termination condition) do
3:     update responsibilities
           $$r'_{ij} = s_{ij} - \max_{v\neq j}\big(a_{iv} + s_{iv}\big),\ i, j = 1, \dots, m \qquad (3.127)$$
4:     determine the final values of responsibilities
           $$r_{ij} = \lambda r_{ij} + (1-\lambda)r'_{ij},\ i, j = 1, \dots, m \qquad (3.128)$$
5:     update availabilities
           $$a'_{ij} = \begin{cases} \sum_{u\neq j} \max\big(0, r_{u,j}\big) \text{ if } i = j \\ \min\big[0, r_{jj} + \sum_{u\notin\{i,j\}} \max(0, r_{uj})\big] \text{ otherwise} \end{cases},\ i, j = 1, \dots, m \qquad (3.129)$$
6:     determine the final availability values
           $$a_{ij} = \lambda a_{ij} + (1-\lambda)a'_{ij},\ i, j = 1, \dots, m \qquad (3.130)$$
7: end while
8: determine indexes
           $$c_i = \arg\max_{1\leq j\leq m}\big(r_{ij} + a_{ij}\big),\ i = 1, \dots, m \qquad (3.131)$$
```

4

Cluster quality versus choice of parameters

In previous chapters we presented various algorithms designed for data clustering. In order to use them efficiently though, we have to solve some basic issues namely:

(a) preparing the data,
(b) establishing the proper number of clusters,
(c) estimating the quality of the applied algorithm,
(d) comparing the results obtained by using alternative algorithms.

In this chapter these issues will be considered.

4.1 Preparing the data

When preparing the data for the analysis it is essential to decide on the set of features used to characterise the data and to apply their transformations, if necessary. Sufficient description of these issues exceeds the book's scope. However, we would like to draw the reader's attention to the importance of this stage of data analysis.

The possibility of using specified algorithms and the quality of results obtained highly depend on the set of features used for describing the real objects. Usually, we would like to operate on an efficient representation, which is a small set of features well distinguishing objects coming from different groups. The majority of the measures of similarity discussed in the paragraph 2.2 lose their discriminant abilities as the dimension of the vectors describing the analysed objects increases. On the other hand, the so called predictive analytics[1] sets the requirement of using the biggest possible set of features in order to make proper predictions. A careful choice of features influences not only the algorithm quality and performance, but also allows for better understanding of the data generating process. A range of various methods applied to choose features is the area of interest for statisticians[2] and for machine learning researchers. A wide range of remarks on this topic can be found in monographs [153], [236], in the paper

[1] cf. e.g. E. Siegel, *Predictive Analytics: The power to predict who will click, buy, lie, or die.* Wiley 2013

[2] we recommend the procedure `VARCLUST` described abundantly in chapter 104 of *SAS/STAT® 13.1 User's Guide.* Cary, NC: SAS Institute Inc.

[152] and in the 18-th chapter of the monograph [164]. A classical reference in this area is [112].

There is a distinction in the literature between feature selection and feature extraction. The former consists in selecting a small subset from a feature set and the latter – in establishing a small subset of *new* (artificial) features. In case of k-means algorithm there are two proven methods of feature extraction: one using random projection[3] and another consisting in SVD. A brief overview of important feature selection methods is presented in the work [182]. In the same a simple feature weighing methodthere was suggested consisting in the generalisation of the objective function, applied in k-means algorithm, of the form

$$J_\beta(U, M, \mathbf{w}) = \sum_{i=1}^{m} \sum_{j=1}^{k} \sum_{l=1}^{n} u_{ij} w_l^\beta (x_{il} - \mu_{jl})^2 \tag{4.1}$$

where $\mathbf{w}$ is a vector whose components represent the weights of particular features, with

$$\sum_{l=1}^{n} w_l = 1,\ 0 \leq w_j \leq 1$$

and β being parameter. When $\beta = 0$, the above function is identical with to function (3.1). The authors suggest assuming $\beta < 0$ or $\beta > 1$.

An attempt to combine feature selection with feature extraction is presented in the work [56].

Having a set of measurements, we often apply normalisation, that is, we transform the data so that the values of the j-th feature belong either to the set $[0, 1]$, or to the set $[-1, 1]$. In turn, standardisation consists in transforming

$$x_j \leftarrow \frac{x_j - \overline{x}_j}{\sigma_j},\ j = 1, \ldots, n$$

where $\overline{x}_j$ is the mean value of j-th feature and σ_j represents its standard deviation.

4.2 Setting the number of clusters

It has been assumed up till now that the number of clusters k is known, which, in fact, is not always the case. The most common method of establishing the proper value of k is using certain partition quality measure, $m(k)$ and establishing such a value of k parameter, which optimises the measure. Nonetheless, it must be emphasized that in the majority of cases the values of quality indicators

[3] cf. A. Blum, "Random projection, margins, kernels, and feature-selection". In: C. Saunders, M. Grobelnik, S. Gunn, and J. Shawe-Taylor, eds. *Subspace, Latent Structure and Feature Selection.* LNCS 3940. Springer Berlin Heidelberg, 2006, 52-68.

depend on concrete data. The fact that for particular data certain indicator allows for establishing the proper number of clusters does not mean that in case of different data it will also indicate the proper value of k. For this reason, various methods are applied, which can be divided into following groups, [373]

(a) Data visualisation. In this approach multidimensional data projection on bi- or tridimensional space is applied. Typical representatives of this direction are principal component analysis (PCA) and multidimensional scaling[4].
(b) Optimisation of some criterion function characterising the properties of mixtures of the probability distributions. E.g. the algorithm EM, discussed in Section 3.2, optimises the parameters θ of the mixture for a given value of k. The value of k for which the quality index attains the optimum value is supposed to be the probable number of clusters. The typical indicators i this domain are
  – Akaike information criterion
$$AIC(k) = \frac{1}{m}\Big(-2(m-1) - n_k - \frac{k}{2} l(\theta)\Big) + 3n_p$$
  where n_k denotes the number of cluster parameters, n_p – total number of parameters being optimised and $l(\theta)$ – likelihood function logarithm.
  – Bayesian criterion
$$BIC(k) = l(\theta) - \frac{n_p}{2} \ln(n)$$
(c) Heuristic methods.

The spectral methods, which are discussed in the next chapter, also provide tools allowing to decide on the number of clusters.

Generally, it must be said that, according to many researchers, determining the proper number of clusters is the basic issue concerning the credibility of cluster analysis results. Excessive number of clusters leads to results which are non-intuitive and difficult to interpret, while too small value of k results in information loss and wrong decisions.

4.2.1 Simple heuristics

One of the simplest rules says that[5]

$$k \approx \sqrt{m/2} \tag{4.2}$$

[4] Cf. e.g. I. Borg, and Patrick JF Groenen. *Modern multidimensional scaling: Theory and applications.* Springer, 2005.

[5] Cf. K. Mardia *et al. Multivariate Analysis.* Academic Press 1979, p. 365.

Another, so called *Elbow Method*[6], consists in examining the fraction of the explained variance in the function of the number of clusters. According to this heuristics we take for k such a number of clusters, that adding yet another cluster increases the fraction only slightly. The decision is based on the diagram: on the coordinate axis one puts subsequent values of k, and on the ordinate axis – the percentage of the explained variance corresponding to them. The point where the curvature flexes is the candidate k value. The percentage of the explained variance is conceived as the ratio of the respective within-group variance to the total of the whole data set. However, one should remember that setting the inflection point is not always unambiguous. Another variant of this method consists in examining the variability of an average cluster radius depending on the number of classes k. The radius r_j of the cluster C_j is defined as the maximum distance of a point from C_j to the prototype $\boldsymbol{\mu}_j$. The mean value of the radii of all clusters is indeed the length of an average radius $\overline{r}$. A typical diagram of dependence of $\overline{r}$ on the number of clusters k is presented in Figure 4.1. Usually, this kind of diagram is plotted for multiples of the value of k, e.g. as indicated in figures below, for $k = 2, 4, 8, \ldots$. If a given value changes slightly in the interval $\{k, 2k\}$, then as the number of clusters we select the value of k^* from this interval [289, section 7.3.3].

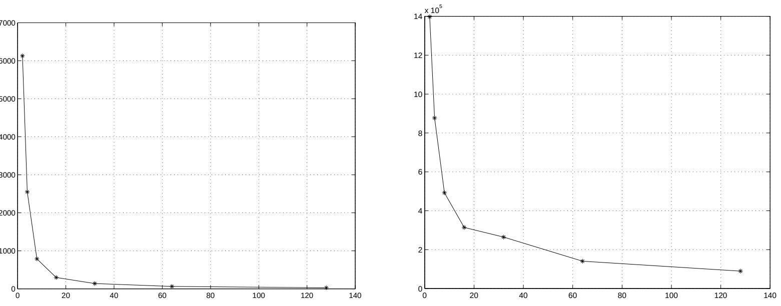


Fig. 4.1. Relation between an average cluster radius (left figure) and the total distance of objects from the prototypes depending on the number of clusters (right figure).

A more formal approach to this problem, together with a survey of other methods can be found in [334]. The authors propose there the so called gap statistics, measuring the change of the within-group variance $W(k, m)$, determined after partition of m-elements set into k clusters (using a chosen clustering algorithm) in relation to the expected variance obtained from an m dimensional sample coming from an exemplary distribution. It is assumed that the value of

[6] The basic work of reference, usually cited while this method is discussed, is the following: R.L. Thorndike "Who Belong in the Family?". *Psychometrika* 18 (4) 1953. Some modification of this method was applied in the paper C. Goutte, P. Toft, E. Rostrup, F.A. Nielsen, L.K. Hansen. "On clustering fMRI time series". *NeuroImage* 9 (3): 298–310 (March 1999).

k, for which the range is maximal, is the probable estimation of the number of clusters.

4.2.2 Methods consisting in the use of information criteria

For determining the value of k, one applies Bayesian information criterion (BIC), already mentioned, Akaike information criterion (AIC), or, finally, the so called deviance information criterion (DIC), which is a generalisation of both previous criteria.

Reference [329] proposes yet another method. For a given partition of the set X into k clusters (knowing their dimensions) one determines an averaged Mahalanobis distance (cf. Section 2.2)

$$\widehat{d}(k) = \frac{1}{n} \min_{j=1,\dots,k} d_{\Sigma}(X, \boldsymbol{\mu}_j) \tag{4.3}$$

where $d_{\Sigma}(X, \boldsymbol{\mu}_j)$ denotes Mahalanobis distance between the elements of the set X and the j-th centre obtained e.g. by using the k-means algorithm.

Subsequently, we determine the range

$$J_k = \widehat{d}^{-\alpha}(k) - \widehat{d}^{-\alpha}(k-1) \tag{4.4}$$

where $\widehat{d}^{-\alpha}(0) = 0$ and $\alpha = n/2$. The value

$$k^* = \arg\max_j J_j \tag{4.5}$$

is assumed to be the proper number of clusters.

In particular, when X is a set without any internal structure, then $k^* = 1$.

In practice, the value of $\widehat{d}(k)$ is approximated by the minimum sum of error squares (3.1), computed by running the k-means algorithm.

4.2.3 Clustergrams

By clustergrams we understand the diagrams in the parallel coordinate system, where a vector was assigned to each observation, [307]. The components of a vector correspond to the cluster membership of the observation for a given number of clusters. The diagrams thus obtained allow for observing the changes in the assignment of objects into clusters in line with the increase of the number of clusters. The author of the idea claims that clustergrams are useful not only in the partitioning methods of cluster analysis but also in hierarchical grouping, when the number of observations increases. At the address `http://www.r-statistics.com/2010/06/clustergram-visualization-and-diagnostics-for-cluster-analysis-r-code/` the code (in the R language) of a programme generating clustergrams is available.

Internal indexes such as the Bayesian information criterion (BIC) and the sum-of-squares encounter the difficulties regarding finding the inflection point of the respective curves and so, detection methods through BIC in partition-based clustering are proposed in the mentioned study. A new sum-of-squares based index is also proposed, where the minimal value is considered to correspond to the optimal number of clusters. External indexes, on the other hand, need a reference clustering or ground-truth information on data and therefore cannot be used in cluster validity. Consequently, an external index was extended into an internal index in order to determine the number of clusters by introducing a re-sampling method.

4.3 Partition quality indexes

It is expected that a clustering algorithm would assign objects which are similar one to another to a common group and those that differ from one another – to different groups. We would like the partition returned by the algorithm, at least for a definite matrix of observations X, to be *optimal* in the sense of the adopted quality index. If the actual assignment of objects into groups is known, the so called *external* quality indexes are constructed. They quantify conformity of the partition returned by an algorithm with the actual assignment of objects into groups. In the opposite case, one attempts to characterise the immanent features of clusters proposed by the algorithm. The basic criteria, applied in constructing *internal* quality indexes belonging to this category, are [156]: *compactness* (groups should be well clustered in the feature space, which means that they should have, e.g., a possibly small diameter) and *separability* (clusters should be easily discernible from each other).

While external indexes allow for selecting the algorithm attaining the highest conformity of the partition with the given partition, internal indexes are used additionally for estimating the number of clusters. A survey of different criteria, to be taken into account during the construction of the external and internal indexes is presented in [161]. Although the work concerns postgenomic research, the notes presented there have general character. In [257], Milligan and Cooper presented a comparison of 30 internal indexes, which are used in hierarchical grouping, while Wu *et al.* outlined in [365] 16 external indexes, which are used for describing k-means algorithm properties. A survey of various quality indexes can be found in [69], [104], [115], [191], [192], [155], [156] and [370]. We recommend also the 3rd chapter of the thesis [388]. Conform to the claim of Kannan, Vempala and Vetta [195], all these indexes, although – because of their simplicity – highly attractive, may lead to wrong conclusions. That is the case mainly when the clusters concerned are not Voronoi clusters (cf. p. 48). This is why the authors quoted, proposed a certain graph-theoretic index of partition quality.

At this point we limit ourselves to measures evaluating the quality of the partition returned by combinatorial algorithms. Milligan and Cooper, already mentioned, compared in [257] 30 different indicators. For this purpose, they constructed testing sets consisting of 2 - 5 clusters containing 50 elements each, described using 4 - 8 features. In their opinion, the best properties are exhibited by the indicator proposed by Caliński and Harabasz, having the form

$$q_{CH}(k) = \frac{tr(B)}{k-1}\frac{m-k}{tr(W)} \tag{4.6}$$

where W and B are matrices of inter- and within-group correlations, defined by the equations (2.27) and (2.28). The optimal number of clusters corresponds to the value of k for which g_{CH} attains the maximum value.

We present below some other often used measures.[7]

It is assumed that the most natural cluster compactness measure is the quantisation error, defined as

$$q(k) = \frac{1}{k}\sum_{j=1}^{k}\frac{1}{n_j}\sum_{\mathbf{x}\in C_j} d(\mathbf{x},\boldsymbol{\mu}_j) \tag{4.7}$$

where $\boldsymbol{\mu}_j$ denotes the cluster centre C_j, n_j denotes its cardinality, while $d(\mathbf{x},\boldsymbol{\mu}_j)$ denotes the distance of the point $\mathbf{x}$ from the center $\boldsymbol{\mu}_j$.

Then, a simple and very popular measure of cluster separation is, [201], the coefficient s_{KR} introduced by Kaufmann and Rousseeuw (called by its authors the *silhouette coefficient*). Let us, namely, denote by $a(\mathbf{x})$ the average distance between some object $\mathbf{x}$ from the class C_j and the rest of objects from this class, and by $b(\mathbf{x})$ – the distance between this point $\mathbf{x}$ and elements of the closest cluster different from C_j:

$$a(\mathbf{x}) = \frac{1}{n_j-1}\sum_{\mathbf{y}\in C_j} d(\mathbf{x},\mathbf{y}) \qquad b(\mathbf{x}) = \min_{l=1,\dots,k,l\neq j}\frac{1}{n_l}\sum_{\mathbf{y}\in C_l} d(\mathbf{x},\mathbf{y})$$

Then

$$s(\mathbf{x}) = [b(\mathbf{x}) - a(\mathbf{x})]/\max[a(\mathbf{x}), b(\mathbf{x})] \tag{4.8}$$

indicates whether the given partition is good (the value of $s(\mathbf{x})$ is close to 1) or bad (negative values of $s(\mathbf{x})$). This indicator is not suitable for large data set analysis; its computational complexity is $O(m^2)$.

The Dunn index, $D(k)$, mentioned in page 98, is used to measure the compactness and the separability of clusters and is defined as

$$D(k) = \min_{j=1,\dots,k}\left\{\min_{l=j+1,\dots,k}\frac{d(C_j,C_l)}{\max_{v=1,\dots,k} diam(C_v)}\right\} \tag{4.9}$$

It is, in effect, a variant of the equation (3.50), where $d(C_j,C_l)$ represents the distance between two clusters, calculated according to the nearest neighbour scheme (in page 35), and $diam(C_v)$ denotes the diameter of the cluster C_v. Its disadvantages are: high computational complexity and sensitivity to outliers. The "optimal" value of k is the one which maximises the index value.

[7] The interested reader should visit the website `http://cran.r-project.org/web/packages/clusterCrit/` which continues to be maintained by Bernard Desgraupes. Description of 42 quality indexes can be found there, as well as clusterCrit package in R, providing their implementation.

Another popular measure is the Davies-Bouldin index, $DB(k)$, defined as follows:

$$DB(k) = \frac{1}{k}\sum_{j=1}^{k} \max_{l=1,\dots,k,l\neq j} \frac{\rho(C_j)+\rho(C_l)}{d(C_j,C_l)} \tag{4.10}$$

where $\rho(C_j)$ denotes the average distance between points from the cluster C_j and the gravity centre of the cluster, that is

$$\rho(C_j) = \frac{1}{|C_j|}\sum_{\mathbf{x}\in C_j} \|\mathbf{x}-\boldsymbol{\mu}_j\|$$

while $d(C_j,C_l)$ denotes the distance between the gravity centres of clusters C_j and C_l. The value of k, for which $DB(k)$ takes on the minimum value, indicates the optimal number of clusters.

Another frequently used quality index, introduced by Xie and Beni [370] has the following form

$$\chi = \frac{\sum_{i=1}^{m}\sum_{j=1}^{k} u_{ij}^{\alpha} d^2(\mathbf{x}_i,\boldsymbol{\mu}_j)}{m\cdot \min_{j\neq l} d^2(\boldsymbol{\mu}_j,\boldsymbol{\mu}_l)} = \frac{J_\alpha(U,M)}{m\cdot \min_{j\neq l} d^2(\boldsymbol{\mu}_j,\boldsymbol{\mu}_l)} \tag{4.11}$$

Here, aspects of both compactness and separability are taken into account. Yet another measure was proposed by Fukuyama and Sugeno [137]:

$$w_{FS} = \sum_{i=1}^{m}\sum_{j=1}^{k}(u_{ij}^{\alpha})[d^2(\mathbf{x}_i,\boldsymbol{\mu}_j) - d^2(\boldsymbol{\mu}_j,\overline{\boldsymbol{\mu}})] \tag{4.12}$$

where $\overline{\boldsymbol{\mu}}$ is the gravity centre of the data set, i.e. $\overline{\boldsymbol{\mu}}_l = (1/m)\sum_{i=1}^{m}\mathbf{x}_{il}$, $l = 1,\dots,n$. The first minimum that the index attains indicates the number of clusters.

Finally, the indicators authored by Gath and Geva [139] are worth mentioning. They suggested two criteria of comparing and selecting the optimal partitioning, assuming that a good partition is characterised by clear separation of clusters, and clustering results contain as many as possible of points centred around the prototype, which means that their volume is minimal. These indicators apply to fuzzy partitions. The first of them, ,,fuzzy volume", has the form

$$F_{HV} = \sum_{j=1}^{c}[det(F_j)]^{1/2}$$

where

$$F_j = \frac{\sum_{i=1}^{N} u_{ij}(\mathbf{x}_i - v_j)(\mathbf{x}_i - v_j)^T}{\sum_{i=1}^{N} u_{ij}}$$

in the situation when the defuzzification parameter is equal to 2. The second indicator, describing the mean density of partition, has the form

$$D_{PA} = \frac{1}{k} \sum_{j=1}^{k} \frac{S_j}{[det(F_j)]^{1/2}}$$

where $S_j = \sum_{i=1}^{N} u_{ij}$. Finally, the density of partition is defined as: $P_D = \frac{S}{F_{HV}}$, where $S = \sum_{j=1}^{k} \sum_{i=1}^{N} u_{ij}$.

4.4 Comparing partitions

Estimating the quality of partition is just one side of the exploratory procedure. As we dispose of many algorithms, we would like to know [354]:

- Is a given algorithm sensitive to outliers (noise)?
- Is the result of algorithm operation dependent on the order in which it processes the data?
- Are the solutions returned by alternative algorithms similar, and if yes, then to what extent?
- Does, for a given task, an optimal solution exist, and if yes, then how close to it are the solutions returned by different algorithms?

The majority of methods of comparing partitions use the contingency table, called also the cross tabulation, the frequency distribution table, or the two way contingency table (in case of comparing two partitions). Let $\mathcal{P}^i = \{C_1^i, \dots, C_k^i\}$, $i = 1, 2, \dots$ denote the partition of the set $\mathfrak{X}$ into k classes returned by i-th algorithm. In particular, it can be assumed that we compare two partitions and each of them divides the set $\mathfrak{X}$ into different number of groups. Without loss of generality, we assume, however, that we are interested in comparing the results of the operation of two algorithms, so $i \in \{1, 2\}$, while each of them divides the set of objects $\mathfrak{X}$ into an established number of classes k. Let

$\mathfrak{X}_{11}$ – set of pairs of objects assigned in both partitions to the same cluster
$\mathfrak{X}_{00}$ – set of pairs of objects assigned in both partitions to different clusters
$\mathfrak{X}_{01}$ – set of pairs of objects assigned in partition $\mathcal{P}^1$ to different clusters and in partition $\mathcal{P}^2$ to the same cluster
$\mathfrak{X}_{10}$ – set of pairs of objects assigned in partition $\mathcal{P}^1$ to the same cluster and in partition $\mathcal{P}^2$ to different clusters
(4.13)

Let, further on, $n_{\alpha\beta} = |\mathfrak{X}_{\alpha\beta}|$. Of course

$$n_{00} + n_{11} + n_{10} + n_{01} = m(m-1)/2$$

and

$$m_j^i = |C_j^i|, \qquad m_{ij} = |C_i^1 \cap C_j^2| \tag{4.14}$$

The measure

$$proj(\mathcal{P}^1, \mathcal{P}^2) = \sum_{C^1 \in \mathcal{P}^1} \max_{C^2 \in \mathcal{P}^2} |C^1 \cap C^2| \tag{4.15}$$

is called the value of projection of the partition $\mathcal{P}^1$ on $\mathcal{P}^2$ [344]. Note that $proj(\mathcal{P}^1, \mathcal{P}^2) \neq proj(\mathcal{P}^2, \mathcal{P}^1)$. The distance $d(\mathcal{P}^1, \mathcal{P}^2)$ between partitions is now defined as

$$d(\mathcal{P}^1, \mathcal{P}^2) = m - proj(\mathcal{P}^1, \mathcal{P}^2) - proj(\mathcal{P}^2, \mathcal{P}^1) \tag{4.16}$$

More remarks on this topic are presented in van Dongen's work [344].

Classical reference here is the work [185]. Referring to [354] and [249], we present a short survey of approaches applied in comparing partitionings. The reader interested in a more thorough discussion should refer to these works and to the bibliographical references [8] quoted there.

4.4.1 Simple methods of comparing partitions

The oldest and the simplest method consists in using the chi-square statistics

$$\chi(\mathcal{P}^1, \mathcal{P}^2) = \sum_{i=1}^{k} \sum_{j=1}^{k} \frac{(m_{ij} - m_i^1 m_j^2)^2}{m_i^1 m_j^2} \tag{4.17}$$

Another method lies in the use of the Rand coefficient

$$\mathcal{R}(\mathcal{P}^1, \mathcal{P}^2) = \frac{2(n_{00} + n_{11})}{m(m-1)} \tag{4.18}$$

which varies from 0 (completely different partitions) to 1 (identical partitions). It is, thus, an equivalent of the recall, i.e. the measure applied in evaluating binary classifiers. The $\mathcal{R}$ coefficient depends on the number of elements and on the number of classes. In particular, if both partitions are independent, then with the increase of k the value of $\mathcal{R}$ tends to 1, which disqualifies this coefficient as a measure of similarity. This is why the so called Adjusted Rand coefficient is introduced, though even that modification is not perfect, [249].

The subsequent indicator is the modified Fowlkes-Mallows coefficient

$$\mathcal{FM}(\mathcal{P}^1, \mathcal{P}^2) = \frac{n_{11}}{\sqrt{(n_{11} + n_{10})(n_{11} + n_{01})}} \tag{4.19}$$

Fowlkes and Mallows introduced their measure in order to compare the results of hierarchical grouping,but its modification, presented above, is used for

[8] Another work of reference is G. Saporta, G. Youness, Comparing two partitions: Some proposals and experiments. *Compstat 2002*, pp. 243-248. It is also worth becoming familiar with the website `http://darwin.phyloviz.net/ComparingPartitions/`.

comparing partitions. As observed in [354], in the context of information retrieval this indicator corresponds to the geometric mean of precision and recall. The measure $\mathcal{FM}(\mathcal{P}^1, \mathcal{P}^2)$ represents a divergence of the ,,empty" model where both partitions are independent. However, a disadvantage of $\mathcal{FM}$ is that for small k it takes on very high values.

The Mirkin metric

$$\mathcal{M}(\mathcal{P}^1, \mathcal{P}^2) = \sum_{i=1}^{k} (n_i^1)^2 + \sum_{j=1}^{k} (n_j^2)^2 - 2 \sum_{i=1}^{k} \sum_{j=1}^{k} m_{ij}^2 \tag{4.20}$$

denotes (the non-normalised) Hamming distance between vectors representing the partitions being compared. The particular elements of vectors correspond to pairs of objects being compared (i, j) and if both objects belong to the same cluster, then we assign 1 to the proper position, and in the opposite case we assign 0,[344]. Furthermore, the following relation exists

$$\mathcal{M}(\mathcal{P}^1, \mathcal{P}^2) = m(m-1)[1 - \mathcal{R}(\mathcal{P}^1, \mathcal{P}^2)] \tag{4.21}$$

4.4.2 Methods measuring common parts of partitions

One of the simplest and the most commonly used measures from this category is the purity (*purity*)

$$\mathfrak{P}(\mathcal{P}^1, \mathcal{P}^2) = \frac{1}{m} \sum_{i=1}^{k_1} \max_{1 \le j \le k_2} m_{ij} \tag{4.22}$$

where k_1 (resp. k_2) denotes the number of groups in the partition $\mathcal{P}^1$ (resp. $\mathcal{P}^2$), and $m_{ij} = |C_i^1 \cap C_j^2|$. We introduced this measure in section 3.3.3.

A subsequent measure, commonly used in information retrieval, is the so called F-measure, that is, the weighted harmonic mean of the precision and recall [246, p. 156]. Suppose we treat the clusters of the partition $\mathcal{P}^1$ as predefined classes of documents and clusters of the partition $\mathcal{P}^2$ as the results of queries. Then the F-measure of the dependance between cluster C_j^2 and another cluster C_i^1 indicates how well C_j^2 describes the class C_i^1

$$F(C_i^1, C_j^2) = \frac{2re_{ij}pe_{ij}}{re_{ij} + pr_{ij}} = \frac{2m_i^1 m_j^2}{m_i^1 + m_j^2} \tag{4.23}$$

where $re_{ij} = m_{ij}/m_i^1$ denotes the recall (*recall*) and $pr_{ij} = m_{ij}/m_j^2$ the precision (*precision*). The complete F-measure is defined as weighted sum of

$$F(\mathcal{P}^1, \mathcal{P}^2) = \frac{1}{m} \sum_{i=1}^{k} m_i^1 \max_{j=1,\dots,k} F(C_i^1, C_j^2) \tag{4.24}$$

This is an asymmetric measure, so it is appropriate e.g. for comparing how well the obtained partition $\mathcal{P}^2$ approximates the actual partition $\mathcal{P}^1$. The only problem is that usually the actual partition is not known. Meilă [249] mentions a modification, introduced by Larsen

$$F_L(\mathcal{P}^1, \mathcal{P}^2) = \frac{1}{k} \sum_{i=1}^{k} \max_{j=1,\dots,k} F(C_i^1, C_j^2) \tag{4.25}$$

Yet, the authors of [354] note that Larsen did not apply such modification.

Another asymmetric measure

$$MH(\mathcal{P}^1, \mathcal{P}^2) = \frac{1}{m} \sum_{j=1}^{k} \max_{i=1,\dots,k} m_{ij} \tag{4.26}$$

was proposed by Meilă and Heckerman in [250]. Again, $\mathcal{P}^1$ is an exemplary partition. In order to adapt this measure for comparing arbitrary partitions the authors introduced a symmetric measure having the form

$$MH_m(\mathcal{P}^1, \mathcal{P}^2) = \frac{1}{m} \sum_{j=1}^{k} m_{ij^*} \tag{4.27}$$

where m_{ij^*} is determined as follows. Let M_1 denote the original contingency table with entries m_{ij} as defined above. In a matrix M_j let m_{ij^*} be a cell with coordinates $(i(j^*), j^*)$ holding a maximum entry of M_j, that means these coordinates mean that the cluster $C_{j^*}^2$ matches the best the cluster $C_{i(j^*)}^1$. M_{j+1} is obtained them by "deleting" the corresponding line $(i(j^*))$ and column (j^*) from the contingency table M_j (e.g. by setting them to -1). While in previous definitions the partitions could have different number of clusters, in this one they should have the identical numbers of clusters.

Then, Wallace [356] postulated introducing an index, representing the probability that a pair of objects taken from a certain cluster $C_i^1 \in \mathcal{P}^1$, in the partition $\mathcal{P}^2$ also belongs to the same cluster

$$WI(\mathcal{P}^1, \mathcal{P}^2) = \frac{M}{\sum_{i=1}^{k} m_i^1 (m_i^1 - 1)/2} \tag{4.28}$$

Here $m_i^1 = |C_i^1|$ and M denotes the number of all the pairs of objects which belong to the same cluster in both partitions. If both partitions are identical, then $WI(\mathcal{P}^1, \mathcal{P}^2) = 1$, and if they are completely different, i.e. $C_i^1 \cap C_j^2 = \emptyset$ for all (i, j), then $WI(\mathcal{P}^1, \mathcal{P}^2) = 0$.

4.4.3 Methods using mutual information

Let m_j denote the cardinality of the j-th cluster $C_j \in \mathcal{P}$ and let $p_j = m_j/m$ denote the probability that a randomly selected element $\mathbf{x} \in X$ belongs to that cluster. The entropy of the partition $\mathcal{P}$ is calculated as

$$H(\mathcal{P}) = -\sum_{j=1}^{k} p_j \log p_j \tag{4.29}$$

In case of two partitions, mutual information between the partitions equals

$$I(\mathcal{P}^1, \mathcal{P}^2) = \sum_{i=1}^{k}\sum_{j=1}^{k} p_{ij} \log \frac{p_{ij}}{p_i^1 p_j^2} \tag{4.30}$$

where $p_{ij} = m_{ij}/m$ and m_{ij} denotes the number of objects belonging to the class $C_i \in \mathcal{P}^1$ and to the class $C_j \in \mathcal{P}^2$. Thus, I can be calculated as

$$I(\mathcal{P}^1, \mathcal{P}^2) = H(\mathcal{P}^1) - H(\mathcal{P}^1|\mathcal{P}^2) \tag{4.31}$$

where the conditional entropy is defined as

$$H(\mathcal{P}^1|\mathcal{P}^2) = -\sum_{i=1}^{k}\sum_{j=1}^{k} p_{ij} \log \frac{p_{ij}}{p_j^2} \tag{4.32}$$

Even though the mutual information is a metric in the space of all partitions, it is not limited, which gives rise to problems of comparing different partitions. However, the following estimation is true

$$0 \le I(\mathcal{P}^1, \mathcal{P}^2) \le \min \Big[H(\mathcal{P}^1), H(\mathcal{P}^2) \Big] \tag{4.33}$$

Strehl and Ghosh proposed in [327] the normalised mutual information

$$\begin{aligned} NMI_{SG}(\mathcal{P}^1, \mathcal{P}^2) &= \frac{I(\mathcal{P}^1,\mathcal{P}^2)}{\sqrt{H(\mathcal{P}^1)H(\mathcal{P}^2)}} \\ &= \frac{\displaystyle\sum_{i=1}^{|\mathcal{P}^1}\sum_{j=1}^{|\mathcal{P}^2} m_{ij} \log \left(\frac{m m_{ij}}{m_i} m_j \right)}{\sqrt{\left(\displaystyle\sum_i m_i \log \frac{m_i}{m} \right)\left(\displaystyle\sum_j m_j \log \frac{m_j}{m} \right)}} \end{aligned} \tag{4.34}$$

The thus defined measure satisfies the conditions $NMI_{SG}(\mathcal{P}^1, \mathcal{P}^2) = 1$ if $\mathcal{P}^1 = \mathcal{P}^2$ and $NMI_{SG}(\mathcal{P}^1, \mathcal{P}^2) = 0$ if $m_{ij} = m_i^1 m_j^2$ or if $m_{ij} = 0$ for all the pairs (i, j).

Then, Fred and Jain [133] normalised the mutual information via the transformation

$$NMI_{FJ}(\mathcal{P}^1, \mathcal{P}^2) = \frac{2I(\mathcal{P}^1, \mathcal{P}^2)}{H(\mathcal{P}^1) + H(\mathcal{P}^2)} \tag{4.35}$$

Like before, $0 \le NMI_{FJ}(\mathcal{P}^1, \mathcal{P}^2) \le 1$. Furthermore, $NMI_{FJ}(\mathcal{P}^1, \mathcal{P}^2) = 0 \Leftrightarrow NMI_{SG}(\mathcal{P}^1, \mathcal{P}^2) = 0$ and $NMI_{FJ}(\mathcal{P}^1, \mathcal{P}^2) = 1 \Leftrightarrow NMI_{SG}(\mathcal{P}^1, \mathcal{P}^2) = 1$.

On the other hand, in the work [225], the following normalisation is used

$$NMI_{LFK}(\mathcal{P}^1, \mathcal{P}^2) = \frac{H(\mathcal{P}^1) + H(\mathcal{P}^2) - H(\mathcal{P}^1, \mathcal{P}^2)}{H(\mathcal{P}^1) + H(\mathcal{P}^2)} \tag{4.36}$$

where $H(\mathcal{P}^1, \mathcal{P}^2) = -\sum_{ij} p_{ij} \log p_{ij}$ denotes the joint entropy.

A measure, which was the subject of detailed analysis, is the disturbance of information, introduced by Meilă [249]

$$\begin{aligned} VI(\mathcal{P}^1, \mathcal{P}^2) &= H(\mathcal{P}^1) + H(\mathcal{P}^2) - I(\mathcal{P}^1, \mathcal{P}^2) \\ &= H(\mathcal{P}^1|\mathcal{P}^2) + H(\mathcal{P}^2|\mathcal{P}^1) \end{aligned} \tag{4.37}$$

It is a metric in the space of all partitions of the $\mathfrak{X}$ set, and if we consider only the partitions into $k \leq \sqrt{m}$ classes, then

$$\frac{2}{m} \leq VI(\mathcal{P}^1, \mathcal{P}^2) \leq 2 \log k \tag{4.38}$$

When the number of objects m is a multiplication of k^2, then $VI(\mathcal{P}^1, \mathcal{P}^2) = 2 \log k$, while in general, if $k > \log k$ then $VI(\mathcal{P}^1, \mathcal{P}^2) \leq \log m$. The value of $VI(\mathcal{P}^1, \mathcal{P}^2)$ can be calculated in time $O(m + k^2)$, which includes: setting the value of contingency table (in time $O(m)$) and setting of the proper values of VI in time $O(k^2)$.

In the work [198] the following normalisation of the index VI was proposed:

$$VI_{KLN}(\mathcal{P}^1, \mathcal{P}^2) = \frac{1}{2}\left[\frac{H(\mathcal{P}^1|\mathcal{P}^2)}{H(\mathcal{P}^1)} + \frac{H(\mathcal{P}^2|\mathcal{P}^1)}{H(\mathcal{P}^2)}\right] \tag{4.39}$$

It represents the averaged defect of information if we reason about $\mathcal{P}^1$ knowing $\mathcal{P}^2$ and vice versa.

4.5 Cover quality measures

Studies concerning this topic are very rare, although we face the problem of cover while generating fuzzy partitions and while analysing the ensembles in empirical graphs (cf. e.g. [129]). Below we present the preliminary results obtained by Lancichinetti, Fortunato and Kertész [225].

Let us remind that by cover $\mathcal{C}$ of the set $\mathfrak{X}$ with k subsets we understand a family of sets $\{C_1, \dots, C_k\}$ such that

$$\begin{aligned} &(a)\ C_i \neq \emptyset, i = 1, \dots, k \\ &(b)\ \bigcup_{i=1}^{k} C_i = \mathfrak{X} \\ &(c)\ C_i \cap C_j \neq \emptyset \end{aligned} \tag{4.40}$$

The (c) condition means that the sets forming the cover are not disjoint.

Assignment of objects into classes is now represented by the table $T^1 = [t^1_{ij}]_{m \times k}$, where $t^1_{ij} = 1$ if i-th object belongs to j-th class, and $t^1_{ij} = 0$ in the opposite case. Let $\mathbf{t}^1_j$ denote j-th column of the matrix T^1 representing the cover $\mathcal{C}^1$. It can be treated as a realisation of dichotomous random variable X_j having probability distribution

$$P(X_j = 1) = m_j^1/m, \qquad P(X_j = 0) = 1 - m_j^1/m \tag{4.41}$$

The random variable Y_j, representing the assignment of objects to the j-th class in the cover $\mathcal{C}^2$ is defined analogously.

Let us determine, after [225], four probability distributions

$$\begin{array}{l} P(X_i = 1, Y_j = 1) = \frac{1}{m}|C_i^1 \cap C_j^2| \\ P(X_i = 1, Y_j = 0) = \frac{1}{m}(|C_i^1| - |C_i^1 \cap C_j^2|) \\ P(X_i = 0, Y_j = 1) = \frac{1}{m}(|C_j^2| - |C_i^1 \cap C_j^2|) \\ P(X_i = 0, Y_j = 0) = \frac{1}{m}(m - |C_i^1| - |C_j^2| + |C_i^1 \cap C_j^2|) \end{array} \tag{4.42}$$

Let further

$$H(X_i|Y_j) = H(X_i, Y_j) - H(Y_j) \tag{4.43}$$

denote the quantity of information, crucial to conclude about X_i, knowing the distribution of the variable Y_j. In particular, if $C_1^1 = C_{j^*}^2$ then $H(X_i|Y_{j^*}) = 0$ and one can say that Y_{j^*} is the best candidate for concluding about the distribution X_i. Thus, it is assumed that

$$H(X_i|\mathcal{C}^2) = H(X_i|\{Y_1, \dots, Y_k\}) = \min_{j=1,\dots,k} H(X_i|Y_j) \tag{4.44}$$

This measure can be normalised

$$H_{norm}(X_i|\mathcal{C}^2) = \frac{H(X_i|\mathcal{C}^2)}{H(X_i)} \tag{4.45}$$

and we get the final formula

$$H_{norm}(\mathcal{C}^1|\mathcal{C}^2) = \frac{1}{k}\sum_{i=1}^{k} \frac{H(X_i|\mathcal{C}^2)}{H(X_i)} \tag{4.46}$$

For comparing the covers of $\mathcal{C}^1$ and $\mathcal{C}^2$, the arithmetic mean of conditional entropies is applied

$$N(\mathcal{C}^1|\mathcal{C}^2) = 1 - \frac{1}{2}[H_{norm}(\mathcal{C}^1|\mathcal{C}^2) + H_{norm}(\mathcal{C}^2|\mathcal{C}^1)] \tag{4.47}$$

Further information concerning the application and the implementation of the above formula can be found in the appendix to the work [225].

5

Spectral methods in clustering and dimensionality reduction

The classical clustering algorithms, like the k-means, or its fuzzy variant FCM, are computationally efficient because they require comparison of the objects (characterized by feature vectors) with a small set of prototypes (represented by centroids in case of these two algorithms). Their disadvantage is a tacit assumption that the observations were derived from a multivariate normal distribution with sufficiently differentiated mean values and similar covariance matrices. Such an assumption justifies the use of Euclidean (or in general, Minkowski) distance to compare feature vectors. On the other hand, this assumption limits the applicability of such algorithms to cases where the groups are linearly separated, that is the data are located within regions of the space bounded by the "shells".

When such an assumption is no longer valid, relational methods can be more efficient. In these methods we use the notion of similarity or dissimilarity between pairs of objects. A short review of such methods is given in Section 3.3.5.8, and some similarity measures are described in Section 2.2. It is important to note that we can operate with (dis-)similarity also in these cases when the feature space is not a vector space[1].

By propagating transitively similarities from a neighbor to another neighbor we relax the assumption of the distance-based definition of a cluster, and thus the requirement that each cluster must be described by a single prototype[2]. This enables discovering clusters of much richer structure.

However, a disadvantage of relational methods is that we must determine similarity between *all* pairs of objects. As similarity is usually symmetric measure, a sample consisting of m observations requires determination of $m(m-1)/2$ similarities. It is a big challenge, because in many practical problems we need not only storage space, but we must also process similarity matrices of huge size. A solution for the reduction of the number of neighbors is to use a threshold value. If s_{ij} denotes similarity of the objects $\mathfrak{x}_i$ and $\mathfrak{x}_j$, then $N_\tau(\mathfrak{x}_i) = \{\mathfrak{x}_j \in \mathfrak{X}\colon s_{ij} \geq \tau\}$ is the set of objects which are similar to the object $\mathfrak{x}_i$ in degree not lower than a user specified τ threshold. Thereby the similarity matrix becomes sparse. Moreover, by restricting the "neighborhood" of each node, we highlight the fact that the similarity measure has only local meaning, and the size of the neighborhood shows the range in which such a measure is applicable. In particular, when the similarity is (inversely) proportional to Euclidean distance, then the small value

[1] See e.g. H. Lodhi, C. Saunders, J. Shwe-Taylor, N. Cristianini, and C. Watkins. Text classification using string kernels. *J. Mach. Learning and Res.*, **2**:419–444, February 2002.

[2] An example is the Affinity Propagation algorithm from chapter 3.

of this distance implies high similarity. But if the distance is large – it says nothing about the similarity, what shows that in practice the Euclidean distance has only local character, [83].

An important tool, allowing analysis of the data characterized by a similarity matrix is offered by spectral analysis. Like in Sect. 2.5.2, a symmetric matrix $S = [s_{ij}]$ is treated as a generalized adjacency matrix of undirected graph $G = (V, E)$, called similarity graph. The set of nodes V is equivalent to the set of objects, and two nodes are linked together if $s_{ij} \geq \tau$. The weights s_{ij} quantify the strength of a relationships between the two corresponding objects. With such a trick, we can expand the applicability of spectral clustering methods: they allow to extract clusters, which are not only compact, but mainly connected – cf. Fig. 2.7 on p. 52.

Analogously as in case of the graph methods, the problem of partitioning of the set of objects into k groups reduces to the problem of cutting the graph into k connected components. There is a number of methods dedicated to this task – consult e.g. [127] or [305]. In general, the optimal cutting is determined by minimizing or maximizing an objective function. A review of such functions is given in Sect. 5.2.2. However, these optimization tasks are $\mathcal{NP}$-complete. To find an approximate solution we use a relaxation of original task, and these relaxations can be easily solved by using spectral methods (discussed in Sect. 5.2.1). Namely, a similarity graph is embedded into k dimensional Euclidean space, where k stands for the number of clusters, in such a way that the coordinates of its nodes are determined by the entries of eigenvectors of the so-called Laplacian[3], which is nothing but a transformed similarity matrix. Although spectral methods provide approximate solution to the cutting problem, its quality hardly depends on the choice of "appropriate" eigenvectors – we discuss this problem is Sect. 5.2.5.4. Further, it must be stressed that while in the classical tasks the number of connected components to be extracted from a given graph is generally known in advance, in cluster analysis this number must be "discovered". Another problem is that of binarization (or rounding) of the real-valued approximate solution obtained by spectral optimization. In brief, the task of binarization is to allocate the nodes (objects) to corresponding subgraphs (groups).

It must be stressed that the so-called spectral data analysis *exploits* the methods of spectral graph theory, but is not limited solely to the study of graphs.

Spectral methods provide tools to constructively aggregate *local* information in order to penetrate the *global* structure of a set of observations. Moreover, the use of terminology from the spectral graph theory allows for giving a clear interpretation of the introduced concepts. But – what is more important it offers new possibilities of defining clusters.

The empirical graphs, i.e. the graphs used in practical applications, are rather huge. Spectral methods allow to describe important properties of such graphs by analyzing a small number of eigenvectors of a matrix characterizing a given graph. Of course, spectral graph theory is concerned primarily with its own problems, e.g. how to associate a graph to a specific matrix, or which prop-

[3] We introduce this notion in Section 5.2.1, and define it in Appendix C.2.

erties of a graph are precisely covered by such a matrix – see the monograph [75] for a review of theoretical results. But these results can be translated in a straightforward manner into meaningful concepts for data analysis, [78].

Spectral clustering has relatively long tradition – see [346] and the bibliography cited there. In 1973 Donath and Hoffman suggested in their paper [108] to use eigenvectors of adjacency matrix to partition the nodes of a graph. In the same year, Fiedler noted in [122, 123], that the second minimal eigenvalue $\lambda_2 \geq 0$ of the Laplacian L of a graph inform not only about the graph connectivity, but also about the intensity of the connections between the nodes of this graph. Namely:

(a) The number of times 0 appears as an eigenvalue in the Laplacian is the number of connected components in the graph. Thus, the second minimal eigenvalue is positive if and only if G is a connected graph.
(b) The more links between the nodes the grater the value of λ_2. In other words if G is a connected graph spanned over the set of m nodes, then $0 < \lambda_2 \leq m$. In particular $\lambda_2 = m$ if G is complete graph (clique).

In general the λ_2 value dependends on the number of vertices, as well as the way in which vertices are connected. For instance, in Erdös-Rényi random graphs, it decreases with the number of vertices, and increases with the average degree. In free-scale networks (generated by so-called preferential attachment[4] this value scales logarithmically with average degree. See [259] for other properties of spectra of the graph Laplacian.

The value λ_2 is called Fiedler value, and the corresponding eigenvector – Fiedler vector. The elements of this vector can be treated as the degrees of membership in appropriate groups[5] – see Appendix C.2.1.1. Further, the difference between these elements represents a "distance" between appropriate nodes of the graph. This property can be used in graph drawing, and – what is more important – in graphical analysis of the structure of the dataset, represented by a similarity graph, [39]. It suffices to sort the entries of the Fiedler vector and to use the resulting ordering to rearrange rows and columns of the similarity (i.e. generalized adjacency) matrix. Thus, the Fiedler vector seems to be quite useful tool for rearranging rows and columns of similarity matrix, as suggested by Czekanowski in [86] (see p. 22 for this idea). An effect of such a reordering of rows and columns is shown in Fig. 5.1.

Spectral clustering relies upon the use of eigenvectors of a matrix representing a set of observations to partition this set into an appropriate number of groups. The computation of these eigenvectors is not a trivial task, and therefore the spectral clustering is a computationally complex method, although there are ef-

[4] See e.g. A.L. Barabási and R. Albert, Emergence of scaling in random networks, *Science*, **286**(5439): 509-512, 1999.

[5] The values of Fiedler vector, as real numbers, may be positive, negative, or zeros. Hence it is assumed that the nodes of the graph are partitioned into two groups: positive and negative.

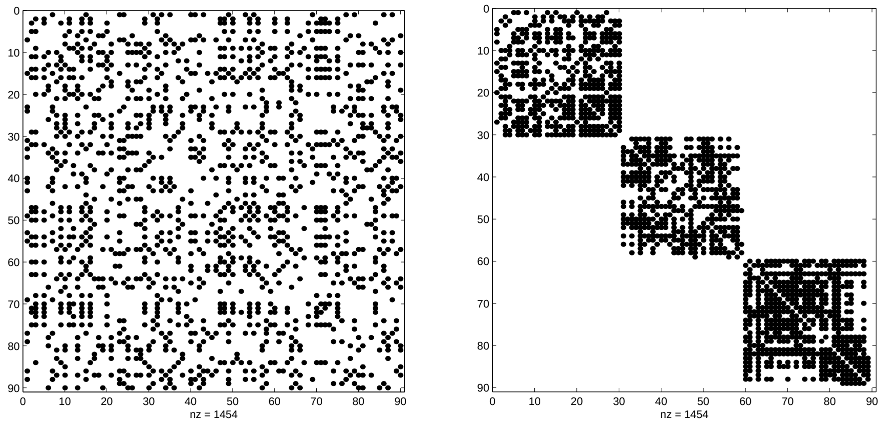


Fig. 5.1. An application of Fiedler vector to visualization of the structure of a dataset. Left panel shows nonzero values of similarity. Horizontal axis: column identifiers of similarity matrix, vertical axis: row identifiers. Black dots: non-zero values of similarity between objects identified by the respective row and columns. Right panel shows the same similarity matrix after the rows and columns have been reordered according to the sorted values of the Fiedler vector, computed for the laplacian $L = \mathrm{diag}(S\mathbf{e}) - S$.

fective methods of calculating the Fiedler vector [245]. One of the first papers, devoted to the use of eigenvectors of the Laplacian of undirected graph was published in 1970 by Hall [157]. An important paper describing practical application of such an approach is that of Pothen, Simon and Liou, [285]. At present, one of most popular papers is the study by Shi and Malik [311], in which an application of spectral clustering to image segmentation is described. The algorithm proposed by Yu and Shi w [379] is considered to be the best (although it is slightly complicated), and the algorithm described in [101] is very efficient in terms of time and memory usage (although it generates rather rough clusters). It should be stressed that

> This is typical of the problems to which spectral partitioning is usually applied: in most circumstances the network in question does not divide up easily into groups of the desired sizes, but one must do the best one can. For these types of tasks, spectral partitioning is an effective and appropriate tool. ([268])

A reader should note that there is no any canonical spectral clustering algorithm. The algorithms belonging to this groups act according to a general 3-stage procedure:

(a) Compute a similarity matrix S describing the relationships among the objects belonging to a given dataset.
(b) Compute a number, say k, of eigenvectors of the matrix S or its transformation (usually a variant of the combinatorial Laplacian derived from this

S). Form a matrix $Y \in \mathbb{R}^{m \times k}$ by stacking the eigenvectors in columns. Rows of this matrix represent "spectral coordinates" of the nodes of the graph corresponding to the matrix S.

(c) Apply a cheap clustering algorithms, e.g. k-means, to cluster the objects described by these new coordinates.

Let us note that even if there does not exist metric representation of the elements of the set $\mathfrak{X}$, after step (b) each element of this set is transformed into a point in k dimensional Euclidean space. Hence, this step may be called "determination of a spectral mapping" $\Phi \colon \mathfrak{X} \to \mathbb{R}^k$.

The main differences about various instantiations of this procedure are concerned with the ways of transforming similarity matrix S. A review of different approaches to spectral clustering can be found in [193], [265] or in Section 5.4.3 of [305], and application of this idea in data analysis is discussed in [248]. Applications of spectral methods in documents clustering and information retrieval are presented in [172] and [173], while applications in bioinformatics can be found in [174]. The tutorial [351] provides an exhaustive review of various methods developed within spectral clustering.

The spectral mapping constructed in step (b) of the general procedure highlights another fascinating feature of spectral approach. It can be used for dimensionality reduction: regardless of the size of the space from which feature vectors originate, after step (b) each such vector is replaced by the k dimensional vector $\mathbf{y}_i$. As noted by Belkin and Niyogi in [43], clustering and dimensionality reduction are two sides of the same coin[6].

Lastly, let us note that spectral methods are resistant to the presence of noise and outliers in data.

5.1 Notation

Spectral graph theory operates with specific notions, see e.g. [75], [351], [320]. That is why we start our presentation from some specific definitions.

Let $\mathfrak{X}$ be a set consisting of m objects. We assume that an information concerning relationships among these objects is given in the form of a similarity matrix S of size m. If the objects can be identified with the vectors $\mathbf{x}_i \in \mathbb{R}^n$, $i = 1, \ldots, m$, and each each vector $\mathbf{x}_i$ represents feature values characterizing object $\mathfrak{x}_i$, then the similarity between each pair of objects is usually computed as follows[7]

$$s_{ij} = \exp\Big(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\omega^2}\Big) = \exp\Big(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2 \Big) \tag{5.1}$$

where $\omega > 0$ is a parameter.

[6] This is only general remark. One can imagine a 2-dimensional set of observations that contain, say 100 clusters. In this case we are faced with "dimensionality explosion"!.

[7] Consult e.g. the paper [43] where a rationale for such a procedure is given.

Referring to the well-known relationship between random walking and diffusion equation, Belkin and Niyogi provide in [43] a justification for such a procedure. On the other hand, Martin says on his blog[8] that the graph laplacian is a low order approximation of the kernel (5.1). This approximation is quite good if i-th and j-th objects are in the same group.

In general, the use of an appropriate formula for evaluating similarity values depends significantly on the data we have at our disposal. A review of various approaches to constructing similarity matrix can be found in Section 3.1 of [193]. Also, one should be conscious, that the choice of a specific formula affects the results of the analysis, see e.g. [243] for a deeper discussion on this subject.

A commonly applied requirement concerning S is that it be a symmetric, and nonnegative matrix. Without loss of generality, we will assume that $s_{ij} \in [0, 1]$ for all pairs $i, j = 1, \ldots, m$.

The symmetric similarity matrix S induces undirected, connected[9] and weighted graph $G = (V, E, S)$ consisting of m vertices. The vertices correspond to the objects, i.e. the symbols $\mathfrak{x}_i$ and v_i refer to the same i-th object from the dataset. A pair of nodes $\{v_i, v_j\}$ is connected by an edge if $s_{ij} \neq 0$. If $\text{nnz}(S)$ stands for number of nonzero matrix S elements, then the graph G has $\mathfrak{m} = \text{nnz}(S)/2$ edges. Each edge $\{v_i, v_j\} \in E$ is equipped with the weight s_{ij} determining similarity, or affinity, of the (connected) objects. We will say that such a graph G is associated with the matrix S, or that G is a similarity graph representing the matrix S.

If the elements of matrix S are computed according to the formula (5.1), we obtain a complete graph consisting of $\mathfrak{m} = m(m-1)/2$ edges. To reduce the number of edges, we usually assume that

$$s_{ij} = \begin{cases} s_{ij} \text{ if } s_{ij} \geq \tau \\ 0 \quad \text{otherwise} \end{cases} \tag{5.2}$$

where $\tau \geq 0$ is a user-defined threshold value. In such a case two nodes are connected only if they are sufficiently similar each to other. Another, commonly used, method relies upon assigning positive wights to k closest neighbors of each vertex. Denoting by $N_k(v_i)$ the set of k closest neighbors[10] of the vertex v_i, we write

$$s_{ij} = \begin{cases} s_{ij} \text{ if } v_j \in N_k(v_i) \\ 0 \quad \text{otherwise} \end{cases} \tag{5.3}$$

More remarks on how to construct similarity graphs can be found in the tutorial [351] or in Appendix E of [229].

When the strength of the similarity is unimportant, we can reduce the similarity matrix to the adjacency matrix A with the elements

[8] `https://charlesmartin14.wordpress.com/2012/10/09/spectral-clustering/`

[9] If the graph is disconnected, i.e. it consists of few independent components, we use the procedures described in this chapter for each component independently.

[10] Equivalently, $N_k(v_i)$ consists of k objects which are most similar to the object v_i.

$$a_{ij} = \begin{cases} 1 \text{ if the nodes } v_i, v_j \text{ are similar} \\ 0 \text{ otherwise} \end{cases} \tag{5.4}$$

It is an extremely simplified similarity matrix (5.2), as it informs only if two objects are similar or not.

Remark 5.1.1 *Formally, adjacency matrix A is a specific kind of similarity matrix. The matrix A characterizes unweighted graph, while the matrix S specifies a graph with weighted edges. The material presented in Section 5.2 concerns both weighted and unweighted graphs. Thus, we will use the similarity matrix S assuming that in some specific situations its elements take the values from the set $\{0, 1\}$. In sections 5.3 and 5.4 we will be concerned, though, with uweighted graphs only. In such cases the symbol A will be used.* □

Definition 5.1.1 *The degree of a vertex $v_i \in V$ is defined as*

$$\mathsf{d}_i = \sum_{j=1}^{m} s_{ij} \tag{5.5}$$

A diagonal matrix D with elements

$$d_{ij} = \mathsf{d}_i \delta_{ij} = \begin{cases} \mathsf{d}_i \text{ if } i = j \\ 0 \text{ otherwise} \end{cases} \tag{5.6}$$

is referred to as the degree matrix . Usually, this matrix is denoted as $D = diag(\mathsf{d}_1, \dots, \mathsf{d}_m)$, as only diagonal elements are of importance. □

The degree matrix can be defined also as follows

$$D = \text{diag}(S \cdot \mathbf{e})$$

where $\mathbf{e}$ stands for the unit vector.

The value $d_{ii} = \mathsf{d}_i$ characterizes total degree of similarity, or affinity, of i-the node (object) to all remaining nodes in a given graph. In fact, if v_i belongs to some cluster, or community, C, we can expect that $s_{ij} \approx 0$ for all nodes v_j outside this cluster. If so, we can suppose that d_{ii} expresses the degree of similarity of the i-th node to all the nodes constituting the cluster C. In other words, $\mathsf{d}_i = d_{ii}$ represents the degree of "typicality", or the degree of "importance" (in the whole set V) of the node v_i. It is natural to assume that each node is similar to at least one[11] other node in V; thus $d_{ii} > 0$, $i = 1, \dots, m$, what implies that the degree matrix is nonsingular.

Definition 5.1.2 *The sum of the weights of all edges attached to vertices belonging to a subset in $Z \subseteq V$ is referred to as the volume of the set Z; it is denoted as $vol Z$, i.e.*

$$vol Z = \sum_{\substack{v_i \in Z \\ v_j \in V}} s_{ij} = \sum_{v_i \in Z} \mathsf{d}_i \tag{5.7}$$

[11] Otherwise such a node will be isolated, or extremely outstanding.

□

In particular

$$\mathrm{vol}\{v_i\} = \sum_{v_j \in V} s_{ij} = \mathsf{d}_i \tag{5.8}$$

is the volume of the node v_i identical to its degree d_i, and

$$\mathrm{vol}\, V = \sum_{\substack{v_i \in V \\ v_j \in V}} s_{ij} = \sum_{i=1}^{m} \mathsf{d}_i \tag{5.9}$$

If S is the adjacency matrix, then

$$\mathrm{vol}\, V = 2\mathfrak{m} \tag{5.10}$$

where $\mathfrak{m}$ stands for the number of edges in the graph G.

Definition 5.1.3 *The pair $\{C, \overline{C}\}$, where $C \subset V$ and $\overline{C} = V \backslash C$ is said to be the graph cut. The cost of this cut is*

$$cut(C, \overline{C}) = R(C, \overline{C}) = \sum_{\substack{v_i \in C \\ v_j \in \overline{C}}} s_{ij} \tag{5.11}$$

□

A subset of edges whose removal splits the graph is denoted ∂C, i.e.

$$\partial C = \{\{i,j\} \in E |\ v_i \in C\ \wedge\ v_j \in \overline{C}\} \tag{5.12}$$

Since S is symmetric, then

$$\partial C = \partial \overline{C} \tag{5.13}$$

The set ∂C is said to be edge separator: removing the edges from ∂C we cut the graph G into two subgraphs: $G_C = (C, E_C)$ and $G_{\overline{C}} = (\overline{C}, E_{\overline{C}})$, where for instance, $E_C = E \cap (C \times C)$. This fact suggests another definition of a cluster (community): we are looking for such a subset of vertices C which only rarely communicate with the vertices belonging to the complement $\overline{C}$. In other words, the majority of neighbors of any node from C also belongs to this set. Such an interpretation is especially meaningful when the clusters differ in their density.

5.2 Spectral data analysis

5.2.1 Spectral optimization

To make the rest of this chapter easy accessible, we start from presenting the basic idea of spectral optimization oriented towards dividing the set of nodes of undirected graph $G = (V, E)$ into $k \geq 2$ disjoint subsets[12], $C_1, \ldots, C_k$, in such a way that the number of links among these subsets is minimal.

5.2.1.1 The case of two classes

Let us start from the simplest case, when $k = 2$, i.e. we are looking for an optimal partition $\{C, \overline{C}\}$. Let S be a similarity matrix, and $G = (V, E)$ be the graph associated to this matrix. Denote $D = \text{diag}(S\mathbf{e})$ the degree matrix corresponding to S. Further, assume that the vector $\boldsymbol{\chi} = (\chi_1, \ldots, \chi_m)^{\mathrm{T}}$ specifies the membership of the nodes in the sets C and $\overline{C}$, i.e.

$$\chi_i = \begin{cases} +1 \text{ if } v_i \in C \\ -1 \text{ if } v_i \in \overline{C} \end{cases} \qquad i = 1, \ldots, m \tag{5.14}$$

It is easy to verify that:

(a) $\chi_i^2 = 1$,
(b) $\boldsymbol{\chi}^{\mathrm{T}}\boldsymbol{\chi} = m$, and
(c) the expression $\frac{1}{4}(\chi_i - \chi_j)^2$ takes the value 1 only when the two nodes belong to different groups; otherwise it is equal zero. In other words, $\frac{1}{4}(1 - \chi_i\chi_j)^2 = 1 - \delta(c_i, c_j)$, where c_i denotes the index of the set containing the node v_i, and δ is Kronecker's symbol.

Let us note that

$$\begin{aligned} \boldsymbol{\chi}^{\mathrm{T}} D \boldsymbol{\chi} &= \sum_{i=1}^{m} d_{ii}\chi_i^2 = \sum_{\substack{v_i \in C \\ v_j \in V}} s_{ij} + \sum_{\substack{v_i \in \overline{C} \\ v_j \in V}} s_{ij} \\ &= \Bigg(\sum_{\substack{v_i \in C \\ v_j \in C}} s_{ij} + \sum_{\substack{v_i \in C \\ v_j \in \overline{C}}} s_{ij} \Bigg) + \Bigg(\sum_{\substack{v_i \in \overline{C} \\ v_j \in \overline{C}}} s_{ij} + \sum_{\substack{v_i \in \overline{C} \\ v_j \in C}} s_{ij} \Bigg) \\ &= assoc(C) + assoc(\overline{C}) + 2cut(C, \overline{C}) = \text{vol}\, V \end{aligned}$$

where

$$assoc(Z) = \sum_{v_i, v_j \in Z} s_{ij} = \text{vol}\, Z - cut(Z, \overline{Z}) \tag{5.15}$$

measures the strength of association between the elements belonging to the set Z.

Similarly

[12] This problem is frequently referred to as the k-way partitioning.

$$\begin{aligned}\boldsymbol{\chi}^{\mathsf{T}} S \boldsymbol{\chi} &= \sum_{i=1}^{m}\sum_{j=1}^{m} s_{ij}\chi_i\chi_j \\ &= \sum_{\substack{v_i\in C\\ v_j\in C}} s_{ij} + \sum_{\substack{v_i\in \overline{C}\\ v_j\in \overline{C}}} s_{ij} - 2\sum_{\substack{v_i\in C\\ v_j\in \overline{C}}} s_{ij} \\ &= assoc(C) + assoc(\overline{C}) - 2cut(C,\overline{C})\end{aligned}$$

Therefore

$$\begin{aligned}\boldsymbol{\chi}^{\mathsf{T}}(D-S)\boldsymbol{\chi} &= 4cut(C,\overline{C}) \\ &= \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} s_{ij}(\chi_i-\chi_j)^2\end{aligned} \tag{5.16}$$

The matrix

$$L = D - S \tag{5.17}$$

is so-called combinatorial (or unnormalized) Laplacian of the graph $G=(V,E)$ with the weights matrix S. It is a symmetric matrix with the elements

$$l_{ij} = \begin{cases} \mathsf{d}_i - s_{ii} \text{ if } i=j \\ -s_{ij} \text{ if } i\neq j \text{ and } (v_i,v_j)\in E \\ 0 \text{ otherwise}\end{cases} \tag{5.18}$$

Basic properties of the Laplacian matrix are given in Appendix C.2.1.1.

If $\boldsymbol{\chi}^{\mathsf{T}}\boldsymbol{\chi} = m$, we can rewrite the equation (5.16) in equivalent form

$$\frac{\boldsymbol{\chi}^{\mathsf{T}} L\boldsymbol{\chi}}{\boldsymbol{\chi}^{\mathsf{T}}\boldsymbol{\chi}} = \frac{4}{m}cut(C,\overline{C}) \tag{5.19}$$

The left hand side of this equation defines so called Rayleigh quotient of the Laplacian L. We will denote it $R(L,\boldsymbol{\chi})$.

In fact, the problem of graph cut minimization reduces to the problem of minimization of the Rayleigh quotient, see e.g. [311] or [96]. The corresponding optimization problem takes the form

$$\begin{aligned}&\min \frac{\boldsymbol{\chi}^{\mathsf{T}} L\boldsymbol{\chi}}{\boldsymbol{\chi}^{\mathsf{T}}\boldsymbol{\chi}} \\ &\text{s.t. } \chi_i = \pm 1, i=1,\dots,m, \boldsymbol{\chi}^{\mathsf{T}}\boldsymbol{\chi} = m \\ &\quad |\boldsymbol{\chi}^{\mathsf{T}}\mathbf{e}| = 0 \text{ or } 1\end{aligned} \tag{5.20}$$

The condition $|\boldsymbol{\chi}^{\mathsf{T}}\mathbf{e}| = 0$ or 1 forces the division into groups of similar cardinality.

The problem (5.20) is an $\mathcal{NP}$-complete optimization problem. A naive method of solving such problems relies upon relaxing their assumptions; the resulting solution is later fitted to the original requirements, defined in the assumptions. In our case, this idea can be implemented as follows: minimization of the quadratic form $\boldsymbol{\chi}^{\mathsf{T}} L \boldsymbol{\chi}$ with the constraint $\chi_i = \pm 1$, $i = 1, \ldots, m$ is relaxed to the problem

$$\begin{array}{c} \min \mathbf{x}^{\mathsf{T}} L \mathbf{x} \\ \text{s.t. } \mathbf{x}^{\mathsf{T}} \mathbf{x} = 1, \mathbf{x} \in [-1, 1]^m \end{array} \tag{5.21}$$

Using the identity (5.16) and ignoring other constraints we immediately find a trivial solution of (5.21). It is a constant vector $\mathbf{y} = \mathit{const}$, as in this case $\mathbf{y}^{\mathsf{T}} L \mathbf{y} = 0$. That is why we search for a unit vector with components in the interval $[-1, 1]$. Such a formulation was given by Hall in [157].

Minimization of the function $f(\mathbf{x}) = \mathbf{x}^{\mathsf{T}} L \mathbf{x}$ with the constraint $\mathbf{x}^{\mathsf{T}} \mathbf{x} = 1$ is a standard optimization problem: first we construct the Lagrangian

$$\mathfrak{l}(\mathbf{x}) = \mathbf{x}^{\mathsf{T}} L \mathbf{x} - \lambda(\mathbf{x}^{\mathsf{T}} \mathbf{x} - 1) \tag{5.22}$$

where λ denotes the Lagrange multiplier. The critical values of $\mathfrak{l}$ occur where its partial derivative $\partial \mathfrak{l}(\mathbf{x}) / \partial \mathbf{x}$ equals zero

$$L\mathbf{x} - \lambda \mathbf{x} = 0 \Rightarrow L\mathbf{x} = \lambda \mathbf{x}$$

which implies that the solution to the problem (5.21) is an eigenvector of the Laplacian[13]. Indeed, if $\mathbf{w}_i$ is an eigenvector corresponding to the eigenvalue λ_i of the Laplacian, then

$$\mathbf{w}_i^{\mathsf{T}} L \mathbf{w}_i = \mathbf{w}_i^{\mathsf{T}} (L \mathbf{w}_i) = \lambda_i \mathbf{w}_i^{\mathsf{T}} \mathbf{w}_i = \lambda_i \|\mathbf{w}_i\|^2$$

If, according to the constraint of the problem (5.21), we normalize the eigenvectors, then $\mathbf{w}_i^{\mathsf{T}} L \mathbf{w}_i = \lambda_i$.

Remark 5.2.1 *If* $\mathbf{w}$ *is an eigenvector of* L*, then* $\alpha \mathbf{w}$*,* $\alpha \neq 0$*, is also an eigenvector of* L*. Thus, in the sequel, we will not distinguish between normalized and unnormalized eigenvectors. If required, an eigenvector can always be normalized. Such a convention is used in many papers, see e.g. [351].* □

The sum of elements of each row of the Laplacian is always equal zero. Hence, if $\mathbf{w}_1$ is a normalized constant vector, then $\|\mathbf{w}_1\| = 1$, and $L\mathbf{w}_1 = 0$. Thus, from the equation $L\mathbf{w}_1 = 0 = \lambda_1 \mathbf{w}_1$ it follows that the minimal eigenvalue of the Laplacian[14] is $\lambda_1 = 0$, and – in consequence – the minimum of the quadratic form $\mathbf{x}^{\mathsf{T}} L \mathbf{x}$ is 0. Surely, such a solution is meaningless as it does not provide any guidance as where to perform cutting. In the sequel, the pair $(\lambda_1, \mathbf{w}_1)$ will be called the trivial eigenpair of the Laplacian. This explains the second requirement for the components of the solution.

[13] If the vector $\mathbf{x} \neq \mathbf{0}$ satisfies the equation $L\mathbf{x} = \lambda \mathbf{x}$, where L is a square matrix, then $\mathbf{x}$ is said to be an eigenvector of L, and the scalar λ – eigenvalue of the matrix L. The eigenvector $\mathbf{x}$ is nontrivial if it is different from a constant vector. If L is a real and symmetric matrix, then all its eigenvalues are real numbers, and the eigenvectors corresponding to different eigenvalues are orthogonal. More properties of eigenvalues and eigenvectors are given in Appendix B.3.

[14] Laplacian is positive semi-definite – see Section C.2.1.1 for details.

Let us order increasingly[15] the eigenvalues of the Laplacian: $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_m$. Since the normalized eigenvectors corresponding to different eigenvalues are orthonormal, thus

$$\mathbf{w}_1^{\mathrm{T}}\mathbf{w}_2 = \frac{1}{\sqrt{m}}\mathbf{e}^{\mathrm{T}}\mathbf{w}_2 = \frac{1}{\sqrt{m}}\sum_{j=1}^{m} w_{2,j} = 0$$

hence the eigenvector $\mathbf{w}_2$, corresponding to a *positive* eigenvalue, contains both positive and negative elements. Let us recall that λ_2 is referred to as the Fiedler value, and the corresponding eigenvector – Fiedler vector. Thus we obtain the next

Theorem 5.2.1 *A nontrivial solution to the problem (5.21) is the Fiedler vector* $\mathbf{w}_2$*, and the Fiedler value is the minimal positive value of the quadratic form* $\mathbf{x}^T L\mathbf{x}$. □

Similar reasoning can be applied to a relaxation of the problem (5.20), relying upon replacing the vector $\boldsymbol{\chi}$ by a unit vector $\mathbf{x}$ with real-valued entries. This time one should apply Courant-Fisher theorem (presented as Theorem B.3.1 on page 234). A review of different methods of Rayleigh quotient minimization can be found in [19].

Discretization of a solution of the problem (5.21) can be performed as follows

$$\chi_i = \begin{cases} +1 \text{ if } w_{2,i} \geq 0 \\ -1 \text{ if } w_{2,i} < 0 \end{cases} \tag{5.23}$$

Fiedler proved in [123] that when G is a connected and undirected graph, and the set C (resp. $\overline{C}$) contains the nodes, for which $\chi_i \geq 0$ (resp. $\chi_i < 0$), then the subgraph $(\overline{C}, E_{\overline{C}})$, spanned on the nodes belonging to the set $\overline{C}$, is connected. Similarly, the subgraph spanned on the nodes from the set C is connected if $w_{2,i} > 0$ for all the nodes from the set C. It should be noted that a small Fiedler value implies small value of the $cut(C, \overline{C})$, [9]. This is quite reasonable, as lower Fiedler value testifies to the "weak" graph connectivity, i.e. the graph is easy to cut.

From this presentation we know that the minimal nontrivial (i.e. positive) value of the Rayleigh quotient (also minimal value of the quadratic form $\mathbf{x}^{\mathrm{T}} L\mathbf{x}$) is the Fiedler value of the Laplacian of a connected graph. But the Fiedler vector corresponding to this value is an approximation of the correct solution! Thus, the rule (5.23) does not guarantee optimality of the solution to the problem (5.20). That is why, instead of the "mechanical" strategy (5.23), a thresholded discretization is frequently used. The cut is performed as follows

$$\chi_i(\tau) = \begin{cases} +1 \text{ if } w_{2,i} \geq \tau \\ -1 \text{ if } w_{2,i} < \tau \end{cases} \tag{5.24}$$

The greedy variant of this strategy relies upon the decreasing ordering of values of the Fiedler vector, and choosing as τ subsequent unique values from this

[15] As already mentioned, these eigenvalues are non-negative real numbers.

ordering. The threshold providing minimal value of the Rayleigh quotient is taken as optimal. Pseudocode 5.1 illustrates a naïve implementation of such a recipe.

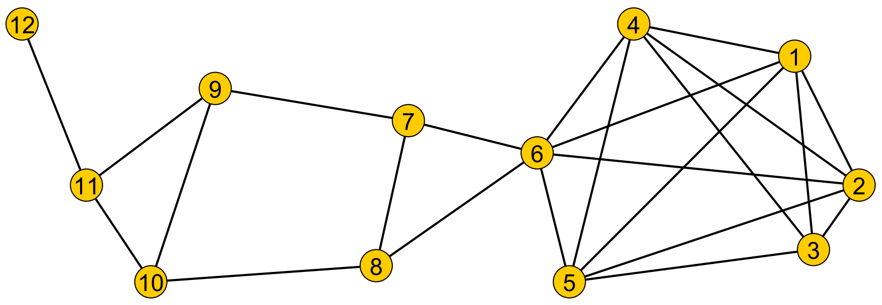


Fig. 5.2. An exemplary graph

Example 5.2.1 *Consider the graph, presented in Fig. 5.2. Assuming that S is identical with the adjacency matrix A of this graph we compute Laplacian $L = \mathrm{diag}(A\mathbf{e}) - A$. Since G is connected, the second minimal eigenvalue is positive, and the Fiedler vector has the form*

$$\mathbf{w}_2 = \begin{array}{c} x \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \\ 12 \end{array} \begin{array}{c} y \\ \left[\begin{array}{r} 0.3147 \\ 0.3147 \\ 0.3147 \\ 0.2766 \\ 0.2766 \\ 0.0951 \\ -0.0979 \\ -0.0979 \\ -0.2672 \\ -0.2672 \\ -0.3717 \\ -0.4906 \end{array}\right] \end{array}$$

When $\tau = 0$, we obtain balanced partition $C_0 = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_6\}$, $\overline{C}_0 = \{\boldsymbol{x}_7, \dots, \boldsymbol{x}_{12}\}$ with Raileigh quotient value $8/12$. But its minimal value $4/12$ will be obtained if we assume $\tau = -0.3717$, what leads to the unbalanced partition $C = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_{11}\}$, $\overline{C} = \{\boldsymbol{x}_{12}\}$. Please note, that by multiplying these values by the factor $4/12$ we obtain, in accordance with the equation (5.19), the cut cost equal 2 in the first case and 1 in the second. □

Algorithm 5.1 Algorithm of thresholded discretization of the Fiedler vector

```
 1: Compute the Fiedler vector $\mathbf{w}_2$ for Laplacian $L$
 2: Determine unique threshold values $\{\tau_1, \dots, \tau_M\}$, $M \le m$
 3: for $i = 1$ to $M$ do
 4:    for $j = 1$ to $m$ do
 5:       if $(w_{2,j} \ge \tau_i)$ then
 6:          $\chi_j = +1$
 7:       else
 8:          $\chi_j = -1$
 9:       end if
10:    end for
11:    Compute the value $R_i = \boldsymbol{\chi}^{\mathsf{T}} L \boldsymbol{\chi} / (\boldsymbol{\chi}^{\mathsf{T}} \boldsymbol{\chi})$
12: end for
13: return cut $\{C, \overline{C}\}$ corresponding to vector $\boldsymbol{\chi}_i$ guaranteeing minimal value of $R_i$
```

The idea of thresholded discretization was used e.g. by Wu and Leahy [369], and by Hagen and Kahng in [154] (in this case the authors used slightly modified quality measure). A more elaborated approach to this problem has been proposed by Tolliver and Miller in [336]. Barnard and Simon described in [40] an interesting approach designed for the analysis of large data. To cut the graph into k components these authors bisect iteratively the subsequent subgraphs. They noted that the algorithm 5.1 favors cutting small subsets of isolated nodes. The methods of avoiding such problems are discussed in Section 5.2.3.

5.2.1.2 Further applications of the Fiedler vector

Fiedler vector and Fiedler value play important role in spectral graph theory[16]. Here below only two potential applications of these notions are presented. The first one provides yet another interpretation of the Fiedler vector. Also, it is a basis for a method of drawing graphs. The second, more important for us, provides a method for assessing the number of clusters in a given dataset.

Hall, in his paper [157] formulates the following problem:

> Given n points (or nodes) and an $n \times n$ symmetric connection matrix, $C = (c_{ij})$, where $c_{ii} = 0$, and $c_{ij} \ge 0$, $i \ne j$, $i = 1, 2, \dots, n$, is the "connection" between point i and point j, find locations for the n points which minimizes the weighted sum of squared distances between the points (i.e., weighted by c_{ij}).

In our setting we have m points and instead of the square matrix C we use the similarity matrix S. Hall considers the one-dimensional problem, that is – he searches for a vector $\mathbf{y} = (y_1, \dots, y_m)$ that minimizes the weighted sum of squared distances between the points. This weighted sum is defined as follows

[16] See e.g. J. Demmel: "CS 267: Notes for Lecture 23, April 9, 1999. Graph Partitioning, Part 2.", `http://www.cs.berkeley.edu/~demmel/cs267/lecture20/lecture20.html`..

$$f(\mathbf{y}) = \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} s_{ij}(y_i - y_j)^2 \tag{5.25}$$

where y_i denotes the 1-dimensional coordinate of point i, and s_{ij} is the connection strength of the pair i and j. It is obvious that if $\mathbf{y}$ is a constant vector, in particular $y_i = 0$ for all i, then $f(\mathbf{y}) = 0$. To avoid such a trivial situation, Hall introduces the constraint $\mathbf{y}^{\mathrm{T}}\mathbf{y} = 1$.

By applying the reasoning from the previous section (and inspired by Hall's paper) we state that the solution to the problem (5.25) is the eigenvector $\mathbf{w}_2$ corresponding to the Fiedler value of the Laplacian $L = D - S$. It determines the "spectral coordinates" of the nodes projected on the real axis. A result of such a projection, applied to the graph from Fig. 5.2, is presented in Fig. 5.3(a). The node numbers are on the abscissa, and the values of the Fiedler vector are placed on the ordinate[17]. More remarks on graph drawing and spectral graph theory can be found e.g. in [320].

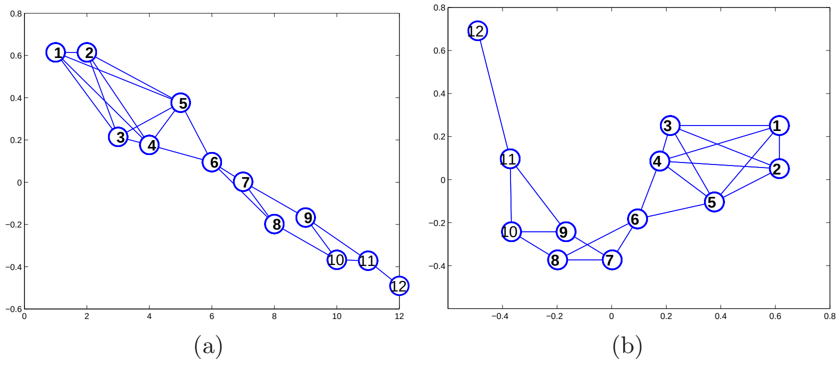


Fig. 5.3. Graph from Fig. 5.2 in spectral coordinates: (a) one-dimensional projection, (b) two-dimensional projection determined by the eigenvectors $\mathbf{y}_2$ (abscissa) and $\mathbf{y}_3$ (ordinate)

Now, let us briefly describe second application. If there are k well defined clusters in the dataset, then the similarity matrix will have (after appropriate rearrangement of rows and columns) the block diagonal structure, i.e. $s_{ij} > 0$ if the objects i and j belong to the same cluster, and $s_{ij} \approx 0$ otherwise. The Laplacian $L = \text{diag}(S\mathbf{e}) - S$, corresponding to this matrix, will also be block diagonal. In such a case the set of eigenvalues of this Laplacian is the union of the eigenvalues of the blocks. The minimal eigenvalue of each block is $\lambda_1^{(j)} \approx 0$, $j = 1, \ldots, k$, hence the first k minimal eigenvalues of the Laplacian will be close

[17] For the nodes v_1, v_2, v_3 the values of this vector are identical. Similar situation occurs in case of the nodes v_4, v_5. For clearer visualization we added small random numbers from the interval $[-0.3, 0.1]$ to the components of this vector.

to zero, and the next, $(k+1)$-th eigenvalue will be $\lambda_{k+1} = \min_{1 \le j \le k} \lambda_2^{(j)}$. When L has a well-defined block-diagonal structure, λ_{k+1} must be definitely greater than λ_k. This phenomenon is illustrated in Fig. 5.4. On the left panel 20 first (i.e. smallest) eigenvalues of the Laplacian of the set `data6_2` are depicted. The clear difference between the 6-th and 7-th eigenvalues pretty well corresponds with our supposition. On the right panel the eigenvalues of Laplacian corresponding to the set `2moons`[18] are shown. Two clusters constituting this dataset are not linearly separable, but also in this case two first eigenvalues are smaller than the remaining eigenvalues.

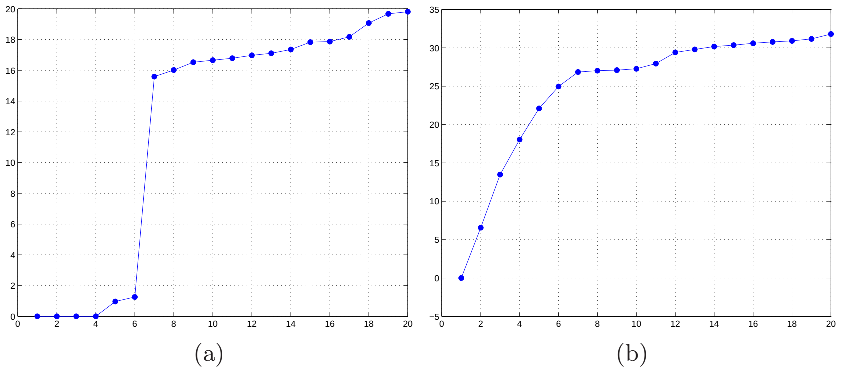


Fig. 5.4. First 20 Laplacian eigenvalues obtained for the sets: (a) `data6_2` and (b) `2moons`. In case (a) $\sigma = 2$ was assumed, and in case (b) $\sigma = 0.6$ was assumed.

5.2.1.3 The multiclass problem

Hall [157] generalizes his reasoning to the case of k-dimensional Euclidean space by introducing the objective function of the form

$$f(\mathbf{y}_1, \ldots, \mathbf{y}_k) = \sum_{j=1}^{k} \mathbf{y}_j^{\mathsf{T}} L \mathbf{y}_j \tag{5.26}$$

and determining its minimum in the presence of additional constraints $\mathbf{y}_j^{\mathsf{T}} \mathbf{y}_j = 1$, $j = 1, \ldots, k$. In this case the Lagrangian takes the form

$$\mathfrak{l}(\mathbf{y}_1, \ldots, \mathbf{y}_k) = \sum_{j=1}^{k} \mathbf{y}_j^{\mathsf{T}} L \mathbf{y}_j - \sum_{j=1}^{k} \alpha_j (\mathbf{y}_j^{\mathsf{T}} \mathbf{y}_j - 1)$$

where α_j stands for j-th Lagrange multiplier.

[18] Both these datasets are characterized in Chapter 6.

Again, the solution to the so stated problem is constituted by the eigenvectors corresponding to the minimal eigenvalues of the Laplacian. Let us arrange increasingly these eigenvalues: $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_m$. Assuming that $\mathbf{y}_j$ cannot be constant vectors, we take as $\mathbf{y}_j$, $j = 1, \ldots, k$, the eigenvector corresponding to the $(j+1)$-th smallest eigenvalue of L.

When $k = 2$, this solution can be interpreted as stabilized location of the nodes placed on an elastic band[19]. In case of cluster analysis, such reasoning leads to the idea of spectral mapping

$$\mathfrak{s}\colon \mathfrak{X} \ni \mathfrak{x}_i \mapsto (y_{i1}, \ldots, y_{ik}) \in \mathbb{R}^k \tag{5.27}$$

which assigns to each object $\mathfrak{x}_i \in \mathfrak{X}$ a point in k-dimensional coordinates, determined by the i-th entries of k eigenvectors of the Laplacian. Belkin and Niyogi term this mapping as Laplacian eigenmap. If $k = 2$ or $k = 3$, such an approach relies upon visualization of the graph (described by the matrix S) in spectral coordinates determined by the eigenvectors. Fig. 5.3(b) presents the graph from Fig. 5.2 in spectral coordinates, determined by the pair of eigenvectors $\mathbf{y}_2, \mathbf{y}_3$, corresponding to the smallest positive eigenvalues $\lambda_2 = 0.2424$, $\lambda_3 = 0.8602$. A reader interested in this last topic is advised to consult [320].

The equation (5.26) can be rewritten as

$$f(Y) = \mathrm{tr}\,(Y^{\mathsf{T}} L Y) \tag{5.28}$$

where $Y = (\mathbf{y}_1, \ldots, \mathbf{y}_k)$ is an $m \times k$ matrix, such that $Y^{\mathsf{T}} Y = \mathbb{I}$. The rows of this matrix determine the location of the vertices of graph G in k-dimensional space.

Please note that if the columns of the matrix $\widetilde{Y} = (\mathbf{y}_1, \ldots, \mathbf{y}_m)$ are eigenvectors of the Laplacian L, and $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_m)$ is the diagonal matrix, containing eigenvalues of L, then $L\widetilde{Y} = \widetilde{Y}\Lambda$. Additionally, if the eigenvectors are normalized, i.e. $\widetilde{Y}^{\mathsf{T}}\widetilde{Y} = \mathbb{I}$, then

$$\widetilde{Y}^{\mathsf{T}} L \widetilde{Y} = \widetilde{Y}^{\mathsf{T}}(L\widetilde{Y}) = \widetilde{Y}^{\mathsf{T}}\widetilde{Y}\Lambda = \Lambda$$

Thus, $\mathrm{tr}\,(\widetilde{Y}^{\mathsf{T}} L \widetilde{Y}) = \mathrm{tr}\,(\Lambda) = \sum_{j=1}^{m} \lambda_j$. As the matrix Y consists of $k < m$ columns, and $\Lambda_k = \mathrm{diag}(\lambda_{(1)}, \ldots, \lambda_{(k)})$ contains the corresponding eigenvalues, then $\mathrm{tr}\,(Y^{\mathsf{T}} L Y) = \mathrm{tr}\,(\Lambda_k)$. Again, $\lambda_{(1)} \leq, \ldots, \leq \lambda_{(k)}$ are the minimal positive eigenvalues of L.

These considerations can be summarized as follows:

Theorem 5.2.2 *Suppose that the graph corresponding to a similarity matrix S is connected. Then, the matrix $Y^* \in \mathbb{R}^{m \times k}$ minimizing the quadratic form $tr(Y^T L Y)$ under the constraint $Y^T Y = \mathbb{I}$ is a matrix, whose columns are k eigenvectors corresponding to the smallest positive eigenvalues of the Laplacian $L = diag(S\mathbf{e}) - S$.* $\square$

[19] A similar idea is the inspiration for the so-called elastic nets. See e.g. R. Durbin and D. Willshaw: An analogue approach to the traveling salesman problem using elastic net method. *Nature*, **326**: 689–691, 1987; H. Ghaziri and I.H. Osman: A neural network algorithm for the traveling salesman problem with backhauls. *Computers & Industrial Engineering*, **44**: 267–281, 2003.

A proof that there does no exist a matrix $Z \neq Y^*$ such that $Z^{\mathrm{T}}Z = \mathbb{I}$ and $\operatorname{tr}(Z^{\mathrm{T}}LZ) < \operatorname{tr}(\Lambda_k)$, can be found e.g. in [67]. Thus, it follows that for any orthogonal matrix $Z \in \mathbb{R}^{m\times k}$ the following inequality holds[20]:

$$\operatorname{tr}(Z^{\mathrm{T}}LZ) \geq \operatorname{tr}(\Lambda_k) \tag{5.29}$$

Until now we have been concerned with eigenvectors corresponding to *smallest* positive eigenvalues of the Laplacian L. Hall notes in [157] that the eigenvector corresponding to the *maximal* eigenvalue of L is a solution to the problem of maximization of the function $f(\mathbf{y})$. Such an approach can be used if the elements of the matrix S represent dissimilarities (instead of similarities). An idea of this kind was used by Aspvall and Gilbert [24] to find a heuristic solution to the graph coloring problem. Seary and Richards, [309], call the eigenvectors corresponding to the minimal eigenvalues *grouping* vectors, and the eigenvectors corresponding to the maximal eigenvalues – *coloring*vectors. Figure 5.5 illustrates the difference between grouping and coloring vectors.

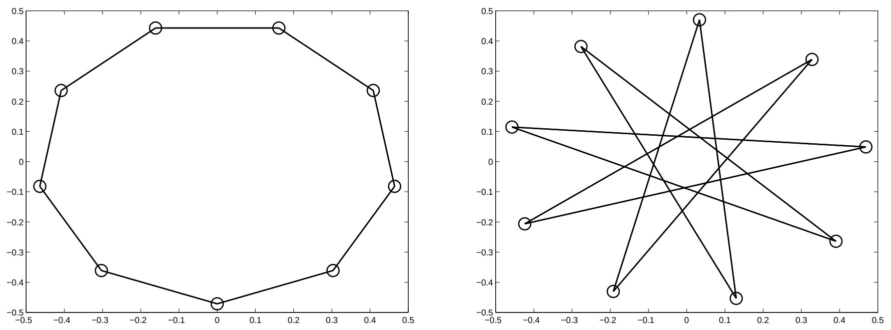


Fig. 5.5. Minimization vs. maximization of the function (5.25). G is the cyclic graph consisting of 9 nodes. Left panel presents the connections between the nodes in the coordinates determined by the grouping vectors (i.e. the eigenvectors corresponding to minimal positive eigenvalues). Right panel illustrates the connections between these nodes in the coordinates determined by the coloring vectors (i.e. the eigenvectors corresponding to largest eigenvalues of the Laplacian L).

5.2.2 Other cutting criteria

Minimization of the cut cost is only one among the possible criteria of grouping nodes of a graph. The use of this criterion, results frequently in unbalanced clusters, one of which consists of only a few nodes. This phenomenon was illustrated in Example 5.2.1. Other criteria used in spectral clustering are presented below.

[20] Another short proof can be found in M.L. Overton, R.S. Womersley, On the sum of largest eigenvalues of a symmetric matrix. *SIAM J. Matrix Analysis and Appl.* **13**(1), 1992, 41-45.

A reader interested in a more detailed discussion of this subject may wish to consult [105], [237], [351], and [311, Sect. 6.1].

Assume that the data set (or set of nodes) should be split into k disjoint subsets $C_1, \ldots, C_k$. The cost of such a split can be measured in terms of the following objective functions:

$$\begin{aligned}
&\text{(a) } Mcut(C_1, \ldots, C_k)) = \sum_{i=1}^{k} cut(C_i, \overline{C}_i) \\
&\text{(b) } Ncut(C_1, \ldots, C_k)) = \sum_{i=1}^{k} \frac{cut(C_i, \overline{C}_i)}{\operatorname{vol} C_i} \\
&\text{(c) } Rcut(C_1, \ldots, C_k)) = \sum_{i=1}^{k} \frac{cut(C_i, \overline{C}_i)}{|C_i|} \\
&\text{(d) } MinMaxcut(C_1, \ldots, C_k) = \sum_{i=1}^{k} \frac{cut(C_i, \overline{C}_i)}{assoc(C_i)}
\end{aligned} \tag{5.30}$$

where *assoc* is the association measure, defined in the equation (5.15). $Mcut$ is the already defined cut function extended to $k \geq 2$ subsets. $Ncut$, *Rcut* and $MinMaxcut$ stand for, respectively, normalized, ratio, and min-max cost of the cut. These criteria were proposed, respectively, in [311], [154], and [105].

Ding *et al.* state in [105] that if there are well separated clusters in the data, then all these criteria behave equally well. But if clusters are "fuzzy" or they are poorly separated, then the $Ncut$ and $MinMaxcut$ should be preferred. Particularly, this last criterion provides better results when the clusters overlap.

So-called Cheeger cut, [65], has many interesting properties. We distinguish between ratio Cheeger cut

$$RCC(C, \overline{C}) = \frac{cut(C, \overline{C})}{\min\big(|C|, |\overline{C}|\big)} \tag{5.31}$$

and normalized Cheeger cut

$$NCC(C, \overline{C}) = \frac{cut(C, \overline{C})}{\min\big(\operatorname{vol} C, \operatorname{vol} \overline{C}\big)} \tag{5.32}$$

In particular, the following inequalities are true

$$RCC(C, \overline{C}) \leq RCut(C, \overline{C}) \leq 2RCC(C, \overline{C})NCC(C, \overline{C}) \leq NCut(C, \overline{C}) \leq 2(C, \overline{C}) \tag{5.33}$$

A clustering algorithm, which uses the NCC criterion is described in [331].

Similarly, it is possible to define a normalized variant of the association measure:

$$Nassoc(C,\overline{C}) = \frac{assoc(C)}{\mathrm{vol}\,(C)} + \frac{assoc(\overline{C})}{\mathrm{vol}\,(\overline{C})} \tag{5.34}$$

As far as the normalized variants of the cut cost emphasize the connections among the elements from different clusters, the normalized variants of association inform about connections among the elements belonging to the same clusters. Thus, the indices from the first group are useful when data are split into clusters, and these from the second group are useful when a group is created. It can be shown, that, see [311],

$$Ncut(C,\overline{C}) = 2 - Nassoc(C,\overline{C})$$

This means that minimization of the normalized cut is equivalent to maximization of the normalized association measure. We discuss this problem in Section 5.2.6.

In the sequel we will focus on most popular indices, *Ncut* and *Rcut*. The *Rcut* criterion balances the number of elements in the clusters, while *Ncut* balances the volume of the clusters by seeking a compromise between cut and association, [288].

Finally, let us mention conductance, called also isoperimetric number of the cut C; it is defined as

$$\varPhi(C) = \frac{|\{\{v_i, v_j\} \in E | v_i \in C \ \& \ v_j \in \overline{C}\}|}{\min\big(\mathrm{vol}\, C, \mathrm{vol}\, \overline{C}\big)} = \frac{|\partial(C)|}{\min\big(\mathrm{vol}\, C, \mathrm{vol}\, \overline{C}\big)} \tag{5.35}$$

More remarks on this notion are given in Appendix C.2.1.1. The number $\varPhi(C)$ can be interpreted as another version of normalization of the cut, or it can be considered in the probabilistic context as presented in Lemma 5.3.1 on page 192.

As $\partial(C) = \partial(\overline{C})$, we can focus on the sets such that $\mathrm{vol}\, C \le \frac{1}{2}\mathrm{vol}\, V$. In such a case

$$\varPhi(C) = \frac{\partial(C)}{\mathrm{vol}\, C} \tag{5.36}$$

The minimal value of the conductance, denoted $\varPhi_G$,

$$\varPhi_G = \min_{C \subset V} \varPhi(C) \tag{5.37}$$

is referred to as graph conductance, Cheeger constant or isoperimetric constant. We mention only two of its properties (for other properties consult e.g. [75]:

(a) G is a connected graph only if $\varPhi_G > 0$.
(b) If G is an unweighted complete graph, then $\varPhi_G = m/(2(m-1))$, if m is even and $\varPhi_G = (m+1)/(2(m-1))$ if m is an odd number.

Remark 5.2.2 *Some authors call the right hand side of the equation (5.35) "isoperimetric number", and they use the term "conductance" only if the cardinality of a set is replaced by the total weight of elements from this set, i.e.*

$$\Phi(C) = \frac{s(\partial(C))}{\min\left(s(C), s(\overline{C})\right)} \tag{5.38}$$

where $s(C)$ stands for the sum of weighted degrees of the nodes belonging to the set C, and $s(\partial(C))$ is the sum of weights of the links joining nodes belonging to different groups. In particular, if $s(\partial C) = cut(C, \overline{C})$, and $s(C) = vol C$, we obtain the normalized Cheeger cut $NCC(C, \overline{C})$. □

5.2.3 The problem of cutting a graph as a generalized eigenproblem

Grouping the nodes of a graph can be imagined as a process of dividing the set of nodes into k disjoint clusters (groups) $C_1, \ldots, C_k$ in such a way, that the total weight of the links joining the nodes belonging to the same group is high, while the total weight of the links joining the nodes from different groups is low. This corresponds relatively well to the natural intuition according to which the elements from different groups are less similar than the elements assigned to the same group. If all the links have identical weights, the above criterion states that the nodes belonging to the same group communicate with each other more often than the nodes belonging to different groups.

Assume, first that the set of nodes is divided into two groups: C and $\overline{C} = V \backslash C$. The naïve criterion $cut(C, \overline{C})$, defined in (5.11), leads to the groups containing single nodes (see example 5.2.1), and, further, it is sensitive to the noise usually present in analyzed data. To overcome these drawbacks, the normalized criteria (b) – (d) from equation (5.30) are used. In particular, using normalization by the volume, as optimal we take the partition $(C, \overline{C})$ for which the index

$$Ncut(C, \overline{C}) = \frac{cut(C, \overline{C})}{\operatorname{vol} C} + \frac{cut(C, \overline{C})}{\operatorname{vol} \overline{C}} \tag{5.39}$$

takes its minimum. The expression $(\frac{1}{\operatorname{vol} C} + \frac{1}{\operatorname{vol} \overline{C}})$ attains the minimal value when $\operatorname{vol} C = \operatorname{vol} \overline{C}$; thus the partition obtained by minimization of the $Ncut$ leads to the groups with similar volumes.

Let $\boldsymbol{\psi}_C$ be a vector representing the membership of the nodes in the set C, i.e.

$$\psi_C(i) = \begin{cases} 1 \text{ if } v_i \in C \\ 0 \text{ otherwise} \end{cases} \quad i = 1, \ldots, m \tag{5.40}$$

If the membership is described by the vector $\boldsymbol{\chi}$ with entries defined in equation (5.14), then $\boldsymbol{\phi}_C = (1 + \boldsymbol{\chi})/2$ and $\boldsymbol{\phi}_{\overline{C}} = 1 - \boldsymbol{\phi}_C = (1 - \boldsymbol{\chi})/2$. Applying the reasoning from Section 5.2.1.1 we conclude that:

$$\boldsymbol{\psi}_C^{\mathtt{T}} D \boldsymbol{\psi}_C = \sum_{j=1}^{m} \mathsf{d}_j \psi_C^2(j) = \operatorname{vol} C,$$

$$\boldsymbol{\psi}_C^{\mathtt{T}} S \boldsymbol{\psi}_C = \sum_{i=1}^{m}\sum_{j=1}^{m} s_{ij}\psi_C(i)\psi_C(j) = \sum_{i,j\in C} s_{ij} = assoc(C)$$

From this, it follows that $cut(C,\overline{C}) = \boldsymbol{\psi}_C^{\mathtt{T}}(D-S)\boldsymbol{\psi}_C$, so

$$Ncut(C,\overline{C}) = \frac{\boldsymbol{\psi}_C^{\mathtt{T}} L \boldsymbol{\psi}_C}{\boldsymbol{\psi}_C^{\mathtt{T}} D \boldsymbol{\psi}_C} + \frac{\boldsymbol{\psi}_{\overline{C}}^{\mathtt{T}} L \boldsymbol{\psi}_{\overline{C}}}{\boldsymbol{\psi}_{\overline{C}}^{\mathtt{T}} D \boldsymbol{\psi}_{\overline{C}}} \tag{5.41}$$

Determining the partition which minimizes this cost is an $\mathcal{NP}$-hard problem. Like in Section 5.2.1.1 we relax this problem by seeking a solution in the set $[-1,1]^{|V|}$. Shi i Malik, after a number of transformations described in [311, Sect. 2.1], bring the equation (5.41) to an equivalent form

$$Ncut(C,\overline{C}) = \frac{\mathbf{y}^{\mathtt{T}} L \mathbf{y}}{\mathbf{y}^{\mathtt{T}} D \mathbf{y}} \tag{5.42}$$

where $\mathbf{y} = \boldsymbol{\psi}_C - b(1-\boldsymbol{\psi}_C)$, and $b = (\operatorname{vol} C)/(\operatorname{vol}\overline{C})$. The entries of the vector $\mathbf{y}$ take the form

$$y_i = \begin{cases} 1 \text{ if } v_i \in C \\ -b \text{ if } v_i \in \overline{C} \end{cases} \quad i = 1,\ldots,m \tag{5.43}$$

The so defined vector satisfies the additional constraint

$$\mathbf{y}^T D\mathbf{e} = \sum_{j=1}^{m} \mathsf{d}_j y_j = \sum_{v_j\in C} \mathsf{d}_j - b\sum_{j\in\overline{C}} \mathsf{d}_j = 0$$

Thus, an approximate solution to the problem of minimizing the quotient (5.41) is the vector

$$\widetilde{\mathbf{y}} = \arg \min_{\substack{\mathbf{y}\neq\mathbf{0}\\ \mathbf{y}^{\mathtt{T}}D\mathbf{e}=0}} \frac{\mathbf{y}^{\mathtt{T}} L \mathbf{y}}{\mathbf{y}^{\mathtt{T}} D \mathbf{y}} \tag{5.44}$$

The right hand side of the equation (5.42) represents the generalized Rayleigh quotient. To solve the problem (5.44) we must determine an eigenvector $\mathbf{y}$ in the generalized eigenproblem (see next Section)

$$L\mathbf{y} = \lambda D\mathbf{y} \tag{5.45}$$

The entries of the vector $\widetilde{\mathbf{y}}$, solving (5.44), are numbers from the set $[-1,1]$ and in general they do not satisfy the condition (5.43). Thus the entries must be rounded or binarized, i.e. converted to 0 or 1.

Remark 5.2.3 *A similar reasoning can be applied to the Rcut criterion:*

$$Rcut(C,\overline{C}) = \frac{cut(C,\overline{C})}{|C|} + \frac{cut(C,\overline{C})}{|\overline{C}|}$$

This equation can be rewritten in the form

$$Rcut(C,\overline{C}) = \frac{\mathbf{x}^T L\mathbf{x}}{\mathbf{x}^T\mathbf{x}} \tag{5.46}$$

where $\mathbf{x} = \boldsymbol{\psi}_C - \beta\boldsymbol{\psi}_{\overline{C}}$, *and* $\beta = |C|/|\overline{C}|$. *Now, the entries of the vector* $\mathbf{x}$ *have the form:* $x_i = 1$ *if* $v_i \in C$, *and* $x_i = -\beta$ *if* $v_i \in \overline{C}$, *which implies that*

$$\mathbf{e}^T\mathbf{x} = \sum_{i=1}^{m} x_i = |C| - \beta|\overline{C}| = 0$$

Thus, an approximate solution to the problem of minimization of the quotient (5.46) is the vector

$$\widetilde{\mathbf{z}} = \arg\min_{\substack{\mathbf{x}\neq\mathbf{0}\\ \mathbf{x}^T\mathbf{e}=0}} \frac{\boldsymbol{x}^T L\mathbf{x}}{\boldsymbol{x}^T\boldsymbol{x}} \tag{5.47}$$

By the Corollary B.7(b) to the minimax Courant-Fisher Theorem (p. 234), we state, that the approximate cost Rcut is equal to the Fiedler value computed for the Laplacian L. The Fiedler vector $\mathbf{x}_2 = \widetilde{\mathbf{x}}$ *satisfies both the constraints, but its entries belong to the interval* $[-1, 1]$, *and not to the interval* $\{-\beta, 1\}$. *Thus we must round this vector again.* □

These considerations can be generalized to the case of $k > 2$ as follows (consult e.g. [351]). Let $\widetilde{\boldsymbol{\psi}}_j$ be the vector indicating the objects belonging to the group C_j, $j = 1, \ldots, k$; it is defined like in equation (5.40). Further, let $\boldsymbol{\psi}_j$ be the vector of the form

$$\boldsymbol{\psi}_j = \frac{1}{\operatorname{vol} C_j}\widetilde{\boldsymbol{\psi}}_j$$

and let Ψ be the matrix whose columns are the vectors $\boldsymbol{\psi}_j$, i.e. $\Psi = (\boldsymbol{\psi}_1, \ldots, \boldsymbol{\psi}_k)$. Since $\boldsymbol{\psi}_i^{\mathrm{T}} D\boldsymbol{\psi}_j = \delta_{ij}$, the matrix Ψ admits the property $\Psi^{\mathrm{T}} D\Psi = \mathbb{I}$. Thus, we state that

$$\frac{cut(C_j,\overline{C}_j)}{\operatorname{vol} C_j} = \boldsymbol{\psi}_j^{\mathrm{T}} L\boldsymbol{\psi}_j = \left(\Psi^{\mathrm{T}} L\Psi\right)_{jj}$$

and finally

$$NCut(C_1,\ldots,C_k) = \sum_{j=1}^{k} \frac{cut(C_j,\overline{C}_j)}{\operatorname{vol} C_j} = \sum_{j=1}^{k} \left(\Psi^{\mathrm{T}} L\Psi\right)_{jj} = \operatorname{tr}\left(\Psi^{\mathrm{T}} L\Psi\right)$$

In other words, the problem of minimization of the normalized cut, $Ncut$, takes the form

$$\begin{aligned} &\min_{C_1,\dots,C_k} \operatorname{tr}(\varPsi^{\mathsf{T}} L \varPsi) \\ \text{subject to} \\ &\varPsi^{\mathsf{T}} D \varPsi = \mathbb{I} \\ &\psi_{ij} = \begin{cases} 1/\operatorname{vol} C_j & \text{if } v_i \in C_j \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{5.48}$$

Replacing $\varPsi$ by the matrix $Y = D^{1/2}\varPsi$ and relaxing the second constraint, i.e. assuming that ψ_{ij} is a real number we transform the above problem into a much easier form

$$\min_{\substack{Y \in \mathbb{R}^{m\times k} \\ Y^{\mathsf{T}}Y = \mathbb{I}}} \operatorname{tr}(Y^{\mathsf{T}} D^{-1/2} L D^{-1/2} Y) \tag{5.49}$$

Similarly, if $Z = (\mathbf{z}_1, \dots, \mathbf{z}_k)$ is the matrix, whose columns are the vectors of the form $\mathbf{z}_j = \widetilde{\boldsymbol{\psi}}_j / |C_j|$, then – as can be easily verified: $\mathbf{z}^{\mathsf{T}}\mathbf{z} = 1$, i.e. $Z^{\mathsf{T}}Z = \mathbb{I}$, and

$$\frac{cut(C_j, \overline{C}_j)}{|C_j|} = \mathbf{z}_j^{\mathsf{T}} L \mathbf{z}_j = \big(Z^{\mathsf{T}} L Z\big)_{jj}$$

Assuming that z_{ij} can take any real value, we reduce the problem of $RCut(C_1, \dots, C_k)$ minimization to much simpler optimization problem

$$\min_{\substack{Z \in \mathbb{R}^{m\times k} \\ Z^{\mathsf{T}}Z = \mathbb{I}}} \operatorname{tr}(Z^{\mathsf{T}} L Y) \tag{5.50}$$

The main difference between the problems (5.49) and (5.50) consists in the use different form of the Laplacian. In the last case the standard (combinatorial) Laplacian has been used. But in the problem (5.49) so-called normalized Laplacian is used; it is defined as

$$\mathfrak{L} = D^{-1/2} L D^{-1/2} \tag{5.51}$$

Like the combinatorial Laplacian, it is a symmetric matrix with the entries

$$\mathfrak{l}_{i,j} = \begin{cases} 1 - \dfrac{s_{ii}}{\mathsf{d}_i} & \text{if } i = j \\ -\dfrac{s_{ij}}{\sqrt{\mathsf{d}_i \mathsf{d}_j}} & \text{if } i,\, j \text{ are neighboring nodes} \\ 0 & \text{otherwise} \end{cases} \tag{5.52}$$

It can be verified, consult e.g. [351, Prop. 2.3], that for any real vector $\mathbf{w} \neq \mathbf{0}$ the following inequality holds

$$\mathbf{w}^{\mathrm{T}}\mathfrak{L}\mathbf{w} = \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} s_{ij}\Big(\frac{w_i}{\sqrt{\mathsf{d}_i}} - \frac{w_j}{\sqrt{\mathsf{d}_j}}\Big)^2 \tag{5.53}$$

which means that $\mathfrak{L}$ is a semi-definite matrix, i.e. its eigenvalues are nonnegative real numbers. In fact, these eigenvalues belong to the interval $[0, 2]$.

Wit the notation $\mathbb{L}$ for the normalized or combinatorial Laplacian, both the problems can be rewritten in a unified (generalized) form

$$\min_{\substack{T\in\mathbb{R}^{m\times k}\\ T^{\mathrm{T}}T=\mathbb{I}}} \operatorname{tr}(T^{\mathrm{T}}\mathbb{L}T) \tag{5.54}$$

The reader should note that such a problem was alredy considered in Section 2.4.2.1. Minimization of the mean-square error has led to the structurally equivalent problem (2.37) from page 45, where – instead of the Laplacian – another semi-definite matrix was used: the Mercer kernel. This intriguing fact inspires investigation of the equivalence between k-means clustering, kernel clustering and spectral clustering, see e.g. [106], [231], [230] for details.

The entries of the matrix T^*, being the solution to the problem (5.54), are real numbers. To translate them into assignments of objects to the groups, we treat the rows of this matrix as spectral coordinates of the objects. If so, we can use, e.g., the k-means algorithm and we should search for the matrices $U^* \in \mathcal{U}_{m\times k}$ and $M^* \in \mathbb{R}^{k\times n}$, which minimize the function (see equation (2.38) from page 46)

$$J_1 = \|T^* - UM\|_F^2$$

Huang, Nie & Huang[21] proposed another modification of the index J_1, namely

$$J_1' = \|T^* - UR\|_F^2$$

Here, R is an orthonormal matrix (i.e. such that $R^{\mathrm{T}}R = \mathbb{I}$). The experiments conducted by the authors mentioned show that their method is competitive with the k-means algorithm both with respect to time complexity and precision of final results. Another way of rounding the solution can be based on the method proposed by Tolliver, [336].

5.2.4 Methods of solving the generalized eigenproblem

Let us focus on solving the problem (5.54). As usual, assume that the set of nodes is divided into two groups. The generalized Rayleigh quotient, i.e. the right hand side of the equation (5.42), can be rewritten as

[21] J. Huang, F. Nie, and H. Huang: Spectral rotation versus k-means in spectral clustering, *Proc. 27-th AAAI Conf. on Artif. Intell.*, pp. 431-437, AAAI Press 2013. MATLAB code implementing this algorithm can be found at `https://sites.google.com/site/feipingnie/publications`.

$$\frac{\mathbf{y}^{\mathrm{T}} L \mathbf{y}}{\mathbf{y}^{\mathrm{T}} D \mathbf{y}} = \frac{\mathbf{y}^{\mathrm{T}} D^{1/2}\Big(D^{-1/2} L D^{-1/2}\Big) D^{1/2}\mathbf{y}}{(\mathbf{y}^{\mathrm{T}} D^{1/2})(D^{1/2}\mathbf{y})} = \frac{\mathbf{z}^{\mathrm{T}} \mathfrak{L} \mathbf{z}}{\mathbf{z}^{\mathrm{T}}\mathbf{z}} \tag{5.55}$$

where $\mathbf{z} = D^{1/2}\mathbf{y}$.

An eigenvector corresponding to the eigenvalue $\lambda_1 = 0$ of the normalized Laplacian $\mathfrak{L}$ is $\mathbf{z}_1 = D^{1/2}\mathbf{e}$. Indeed

$$\mathfrak{L}\mathbf{z}_1 = \big(D^{-1/2} L D^{-1/2}\big)(D^{1/2}\mathbf{e}) = D^{-1/2}(L\mathbf{e}) = 0$$

Thus, $\mathbf{y}_1 = D^{-1/2}\mathbf{z}_1 = \mathbf{e}$ is the eigenvector corresponding to the minimal eigenvalue in the generalized problem (5.45).

The identity (5.55) implies that instead of minimization of the generalized Rayleigh quotient (5.42), it suffices to consider the quotient $R(\mathfrak{L}, \mathbf{z}) = (\mathbf{z}^{\mathrm{T}}\mathfrak{L}\mathbf{z})/(\mathbf{z}^{\mathrm{T}}\mathbf{z})$. It is known – by Courant-Fisher Theorem – that the next minimal value of the Rayleigh quotient over the set of all vectors $\mathbf{z} \in \mathbb{R}^m$ which are orthogonal to the already defined vector $\mathbf{z}_1$, is $R(\mathfrak{L}, \mathbf{z}_2) = \lambda_2$, where $\mathbf{z}_2$ is the Fiedler vector, and λ_2 is the Fiedler value of the matrix $\mathfrak{L}$. Moreover, as the eigenvectors of this matrix are orthogonal, then

$$\mathbf{z}_2^{\mathrm{T}}\mathbf{z}_1 = (D^{1/2}\mathbf{y}_2)^{\mathrm{T}}(D^{1/2}\mathbf{e}) = \mathbf{y}_2^{\mathrm{T}} D\mathbf{e} = 0$$

where $\mathbf{y}_2$ is the eigenvector corresponding to the second minimal eigenvalue in the generalized problem (5.45).

The following theorem summarizes our considerations:

Theorem 5.2.3 *[311] The solution to the problem (5.44) is the vector* $\mathbf{y}_2 = D^{-1/2}\mathbf{z}_2$*, where* $\mathbf{z}_2$ *is the Fiedler vector of the normalized Laplacian* $\mathfrak{L}$*.* □

When $k > 2$, the solution to the problem (5.54) is the matrix T, whose columns are the eigenvectors corresponding to k minimal eigenvalues of the Laplacian $\mathbb{L}$.

Please, note that the eigenproblem (5.45) can be reduced to a standard eigenproblem if we multiply both sides of the equation (5.45) by the matrix D^{-1}:

$$D^{-1}L\mathbf{y} = \lambda\mathbf{y} \tag{5.56}$$

D is a nonsingular matrix (see p. 149 for a rationale), hence the above equation is well defined. Unfortunately, $D^{-1}L$ is an asymmetric matrix what causes problems in the computation of its eigenvalues and eigenvectors.

Alternatively, normalized Laplacian can be represented in equivalent form as $\mathfrak{L} = \mathbb{I} - D^{-1/2}SD^{-1/2}$. The matrix

$$\mathfrak{L}^d = \mathbb{I} - \mathfrak{L} = D^{-1/2}SD^{-1/2} \tag{5.57}$$

is referred to as the complement of the normalized Laplacian. Its application in clustering is discussed in Section 5.2.5.3. Other interesting interpretations of the matrix $\mathfrak{L}^d$ are discussed in [309] and in [41]. Here we only note one important fact[22]:

Lemma 5.2.1 *If* $(\mu, \mathbf{w})$ *is a solution to the eigenproblem* $D^{-1/2}SD^{-1/2}\mathbf{w} = \mu\mathbf{w}$*, then* $(1-\mu, D^{-1/2}\mathbf{w})$ *is a solution to the generalized eigenproblem (5.45).* $\square$

Numerical methods designed to compute the eigenvectors, and discussed in Appendix B.3.3, are oriented towards finding the eigenvectors corresponding to the largest eigenvalues. Thus, the above property allows easy computation of the requested quantities.

In Section C.2.1.1 the Cheeger inequality

$$\frac{1}{2}\Phi_G^2 \le \lambda_F \le 2\Phi_G \tag{5.58}$$

was mentioned. Here λ_F stands for the Fiedler value of the normalized Laplacian. This inequality implies that in order to find a set of small conductance, we must analyse eigenvectors and eigenvalues of the normalized Laplacian. Low Fiedler value implies low conductance.

Another interesting inequality was proposed by Tolliver in his dissertation [335, p. 32]. Let $\mathbf{w}_F$ be the Fiedler vector in the generalized eigenproblem (5.45) and let

$$nc_{min}(G) = \min_{-1<\tau<1} Ncut(C_\tau, \overline{C}_\tau) \tag{5.59}$$

stand for the minimal value of $Ncut$ obtained by thresholded rounding of the vector $\mathbf{w}_F$ (i.e. a node i belongs to the set C_τ if $w_{F,i} \ge \tau$). Then

$$\frac{1}{2}\lambda_1 \le nc_{min}(G) \le \sqrt{2\lambda_1} \tag{5.60}$$

Example 5.2.2 *Consider again the graph from Fig. 5.2. The entries of the Fiedler vector of the normalized Laplacian are:*

$$\begin{array}{c c} x & y \\ \begin{array}{r} 1\\2\\3\\4\\5\\6\\7\\8\\9\\10\\11\\12 \end{array} & \begin{bmatrix} 0.2898\\0.2898\\0.2898\\0.2718\\0.2718\\0.0053\\-0.2022\\-0.2022\\-0.3601\\-0.3601\\-0.4310\\-0.2707 \end{bmatrix} \end{array}$$

[22] It follows immediately from Lemma B.3.1.

and the Fiedler value is $\lambda_F = 0.0805$.

Thresholded rounding results in six vectors $\mathbf{y}_i$ *with the entries in* $\{-1, 1\}$*. These vectors represent six sets together with their complements:* $C_1 = \{1, \ldots, 3\}$, $C_2 = \{1, \ldots, 5\}$, $C_3 = \{1, \ldots, 6\}$, $C_4 = \{1, \ldots, 8\}$, $C_5 = \{1, \ldots, 10\}$ *and* $C_6 = \{1, \ldots, 11\}$*. The values of Rayleigh quotient* $(\mathbf{y}_i^T \mathfrak{L} \mathbf{y}_i)/(\mathbf{y}_i^T \mathbf{y}_i)$ *computed for these vectors are:* 0.4650, 0.1668, 0.2102, 0.2400, 0.2400 *i* 0.2102*. Thus, the minimal value of the quotient is attained for the vector* $\mathbf{y}_2$*, which represents the partition* $\{C_2, \overline{C}_2\}$*. In this manner we avoid the drawback mentioned in Example 5.2.1.* □

Ending this section let us mention few remarks concerning the approach discussed above.

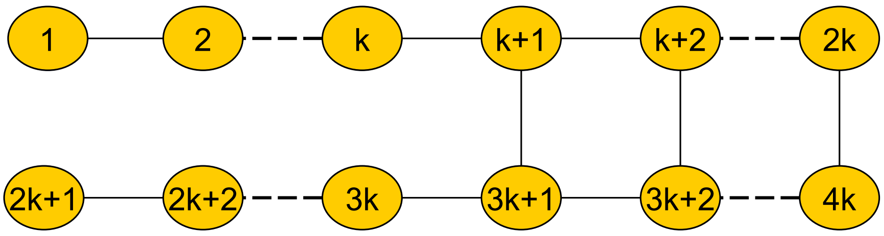


Fig. 5.6. General structure of the graphs analyzed by Guattery and Miller in [148]

Remark 5.2.4 *Guattery and Miller studied in [148] graphs of the form shown in Fig. 5.6. These graphs look like a ladder, with half of rungs removed. Obviously, the best normalized cut (with cost 2) is obtained by removing the initial* k *nodes from each arm of the ladder. But the entries of the Fiedler vector (of both normalized and unnormalized Laplacian) are positive for the nodes numbered from 1 to* $2k$ *and negative for the remaining nodes. This means that the cut should be parallel to the arms of the ladder! Unnormalized cost of such a cut is equal to* k.

A more valuable result can be obtained when we search for the set of lowest conductance. In the graph from Fig. 5.6 we find that the conductance of the set $C_1 = \{\mathbf{x}_1, \ldots \mathbf{x}_k, \mathbf{x}_{2k+1}, \ldots, \mathbf{x}_{3k}\}$ *is* $\Phi(C_1) = 1/(2k-1)$*. If we add to this set two nodes being the ends of the first "rung", i.e.* $C_2 = C_1 \cup \{\mathbf{x}_{k+1}, \mathbf{x}_{3k+1}\}$*, then* $\Phi(C_2) = 1/(3k-4) < \Phi(C_1)$ *for* $k > 3$*. The subgraph spanned over the set* C_1 *of nodes is disconnected (without "rungs"). In the second case the subgaphs spanned over the sets* C_2 *and* $\overline{C}_2$ *are connected, and* $\mathit{cut}(C_2, \overline{C}_2) = 2$. □

Remark 5.2.5 *In practice, the algorithms based on the unnormalized and normalized Laplacian seem to be very similar to each other. But a careful analysis, performed by von Luxburg, Belkin and Bousquet, and presented in [352] shows that in fact these are two different approaches. Particularly, these* *authors state:*

Consistency is a key property of statistical algorithms, when the data is drawn from some underlying probability distribution. Surprisingly, despite decades of work, little is known about consistency of most clustering algorithms. In this paper we investigate consistency of a popular family of spectral clustering algorithms, which cluster the data with the help of eigenvectors of graph Laplacian matrices. We show that one of the two of major classes of spectral clustering (normalized clustering) converges under some very general conditions, while the other (unnormalized), is only consistent under strong additional assumptions, which, as we demonstrate, are not always satisfied in real data. We conclude that our analysis provides strong evidence for the superiority of normalized spectral clustering in practical applications. We believe that methods used in our analysis will provide a basis for future exploration of Laplacian-based methods in a statistical setting.

This leads to two main practical conclusions about spectral clustering. First, from a statistical point of view it is clear that normalized rather than unnormalized spectral clustering should be used whenever possible. Second, if for some reason one wants to use unnormalized spectral clustering, one should try to check whether the eigenvalues corresponding to the eigenvectors used by the algorithm lie significantly below the continuous part of the spectrum. If that is not the case, those eigenvectors need to be discarded as they do not provide information about the clustering.

5.2.5 Algorithms for spectral data clustering

The algorithms belonging to this group exploit the information provided by the eigenvalues and eigenvectors of the similarity matrix or its transformations (i.e. various variants of a Laplacian derived from this matrix).

While the partitional algorithms for cluster analysis, like k-means algorithm, are designed for convex datasets (typically they return the Voronoi decomposition of the entire space), the spectral methods are oriented towards the sets with less rigorous structure. A review of various variants of such algorithms can be found e.g. in [351], [350], or [349].

In spectral clustering either a single eigenvector is used (i.e. a standard bipartition philosophy is applied, see [387] for theoretical foundations of this approach) or a set of k eigenvectors is treated as the set of spectral coordinates, describing location of the data points in the new space. In the first case we speak about the recurrent algorithm, as it is repeated until satisfactory number of groups is obtained. In the second case we speak about spectral mapping. Regardless of the choice of a particular method, the following procedure summarizes the basic steps of spectral clustering:

(a) Determine a similarity matrix S. Its elements describe either a strength of relationship between pairs of objects (usually $s_{ij} \in [0, 1]$), or they only indicate whether such a relationship exists (in this case $s_{ij} \in \{0, 1\}$). Then, the matrix S is transformed to another matrix $\mathcal{M}$; it usually it corresponds to the normalized or unnormalized Laplacian of the graph associated to S.
(b) Compute eigendecomposition of the matrix $\mathcal{M}$, i.e. $\mathcal{M} = W \Lambda W^{-1}$, where W is the square $m \times m$ matrix whose i-th column is the eigenvector $\mathbf{w}_i$ of $\mathcal{M}$ and λ is the diagonal matrix whose diagonal elements are the corresponding eigenvalues of $\mathcal{M}$.
(c) Determine the so-called spectral mapping $X \ni \mathbf{x}_i \mapsto (w_{i1}, \ldots, w_{il})$, i.e. the l-dimensional representation of the n-dimensional dataset. Typically, l equals to the number of groups k (but see also Sect. 5.2.5.4 for an other approach).
(d) Use the low dimensional representation to partition the data.

If $l = 1$, then in step (c) the Fiedler vector is used, and if $l \geq 2$ then to partition the data into k groups the k-means algorithm is frequently used. Different instances of such a procedure, including the most popular ones, are described below. For reader's convenience, in Table 5.1 we briefly characterize these algorithms.

Authors	Matrix	Eigenvectors
Perona & Freeman (1989)	S	Dominating eigenvectors
Scott & Longuet-Higgins (1990)	S	Normalized rows of the matrix $W = (\mathbf{w}_1, \ldots, \mathbf{w}_m)$ and analysis of the matrix $Q = VV^{\mathrm{T}}$.
Shi & Malik (2000)	$D^{-1}(D - S)$	Minimal vectors $\mathbf{w}_2, \ldots, \mathbf{w}_k$
Meilă & Shi	$D^{-1}S$	Rows of the matrix built of k dominating eigenvectors
Ng *et al.* (2004)	$D^{-1/2}SD^{-1/2}$	Normalized rows of the matrix built of k dominating eigenvectors
Shi, Belkin & Yu (2009)	S	k dominating eigenvectors with no sign change, up to precision ϵ.

Table 5.1. Brief characteristics of various algorithms of spectral clustering

The first and second algorithm[23], mentioned in this table were devised for image segmentation. The similarity matrix was computed there by using the Gaussian kernel. Shi and Malik have dealt in [311] with image segmentation, as well. The algorithm proposed in [312] by Shi, Belkin and Yu was designed for clustering of data coming from a Gaussian mixture model. In this last case the

[23] P. Perona and W.T. Freeman: A factorization approach to grouping. In H. Burkardt and B. Neumann (eds.), *Proc ECCV*, LNCS 1407, pp. 655-670, Springer 1998. G.L. Scott and H. C. Longuet-Higgins: Feature grouping by 'relocalisation' of eigenvectors of the proxmity matrix. In: *Proc. British Machine Vision Conference*, University of Oxford, 24-27 Sep. 1990, pp. 103-108.

number of groups is determined by finding the dominating eigenvectors with no sign change (see Sect. 5.2.5.4 for details and explanation).

One should also mention `GRACLUS` algorithm[24], described in [101]. This algorithm is designed for large graphs (constructed e.g. from massive datasets). At the beginning, the entire graph G is reduced (by using appropriate heuristics) to a graph G_t of reasonable size, so that a nested sequence of graphs $G = G_0 \supset G_1 \cdots \supset G_t$ is constructed. Next, the algorithm described in [379] is used to find the minimal cut in G_t. This result is propagated backward *via* intermediate graphs $G_{t-1}, G_{t-2}, \ldots$ to the original graph G.

In the majority of algorithms a symmetric similarity matrix is used. Only Meilă and Pentney described in [251] an algorithm operating with *asymmetric* matrix S. Kleinberg's `HITS` algorithm, [205], belongs to this group too.

All these algorithms try to minimize the indices $MCut$ or $NCut$. Hagen and Kahn considered in [154] the problem of minimization of $RCut$ index; see also the paper by Chan, Schlag and Zien, [67].

From the technical point of view we can use, in some circumstances, the Singular Value Decomposition, instead of spectral decomposition, as described in [271] and [96].

5.2.5.1 SM – The algorithm of normalized cuts

It is one of most popular algorithms, designed for image segmentation and proposed by Shi and Malik in [311]; in the literature it is frequently referred to as the SM algorithm. A characteristic feature of this algorithm is that it does not focus on the external connections between clusters only, but considers, as well, the internal connections within a cluster. So it can produce balanced clustering results.

The authors use the bisection method: at each step they divide the dataset into two subsets until sufficient number of groups is obtained. In a single step the Fiedler vector $\mathbf{w}_2$ of the matrix $\mathfrak{L} = D^{-1/2}(D-S)D^{-1/2}$ is computed, and next the vector $\mathbf{y}_2 = D^{-1/2}\mathbf{w}_2$ is determined. According to our earlier considerations, $\mathbf{y}_2$ is the Fiedler vector of the generalized eigenproblem $L\mathbf{y} = \lambda D\mathbf{y}$. To cut the current piece of data, the rule (5.23) can be applied. But Shi and Malik suggest in [311] that a node v_j is assigned to one group if $y_{2,j} \leq \tau$, and to the other group, otherwise. As the threshold value we can use the median value, following the suggestion from Pothen, Simon and Liou in [285] or $\tau = 0$.

Another method of rounding the vector $\mathbf{y}_2$ relies upon the tentative partition of the dataset and determination of the $NCut$ value for this partition. Next, an increase $\Delta NCut_i$, implied by moving i-th node to the opposite group, is computed. Finally, we choose an element with the number

$$i^* = \arg\min_{1 \leq i \leq m} (NCut + \Delta NCut_i) \tag{5.61}$$

[24] Detailed description of this algorithm and source code are available from `http://www.cs.utexas.edu/users/dml/Software/graclus.html`.

This procedure is repeated until it is impossible to improve the value of $NCut$. It is important to note that at each iteration only single "untouched" node is moved to the opposite group.

After cutting the graph into two subgraphs, one of them can be re-cut. As the new candidate for re-cutting we choose a subgraph with the smallest Fiedler value[25]. Such a commonsense criterion was used by Meilă and Verma in [349].

If a given dataset must be divided into predefined number k of groups, we can construct a matrix U, with columns being k eigenvectors corresponding to the k smallest eigenvalues of the generalized eigenproblem problem $L\mathbf{y} = \lambda D\mathbf{y}$. The rows of this matrix determine spectral coordinates of the nodes (objects). In such a case the k-means algorithm can be used to obtain final partition.

Algorithm 5.2 SM algorithm of normalized cuts, [311]

```
1: Compute similarity matrix $S$; its entries $s_{ij}$ can be computed by using the equation
   (5.1). Compute the degree matrix $D = \mathrm{diag}(S\cdot\mathbf{e})$.
2: Let $\mathbf{w}_2$ be the Fiedler vector of the matrix $\mathfrak{L} = D^{-1/2}(D - S)D^{-1/2}$. Compute the
   vector $\mathbf{y}_2 = D^{-1/2}\mathbf{w}_2$ and use it to partition the set of nodes $V$ into two disjoint
   subsets.
3: If you are not satisfied with the partition, use subsequent eigenvectors to partition
   subgraphs.
4: return Partition of the set $V$ into $k \geq 2$ disjoint subsets.
```

5.2.5.2 VM – another algorithm of normalized cuts

Multiplication of both sides of the equation (5.45) by the matrix D^{-1} leads to the equality $\mathbb{I}\mathbf{y} - D^{-1}S\mathbf{y} = \lambda\mathbf{y}$, which – after introducing $P = D^{-1}S$ and ordering its components – takes the form

$$P\mathbf{y} = \mu\mathbf{y} \tag{5.62}$$

In this manner the generalized eigenproblem was reduced to the standard eigenproblem with $\mu = 1 - \lambda$. P is a row-stochastic matrix, and so its dominating eigenvalue is $\mu_1 = 1$. By sorting decreasingly the eigenvalues of the matrix P we obtain $1 = \mu_1 \geq \mu_2 \geq \cdots \geq \mu_m$, i.e. the Fiedler value corresponds to the second maximal eigenvalue μ_2 (that is $\lambda_2 = 1 - \mu_2$). The reader can verify that the right eigenvectors of the matrix P are identical to the eigenvectors computed in the generalized eigenproblem (5.45). In this way we obtain the algorithm 5.3 equivalent to the SM algorithm 5.2. This new algorithm was proposed by Verma and Meilă in [349]. A reader should note, that while the SM algorithm operates on symmetric similarity matrix S, the VM algorithm uses asymmetric stochastic matrix P. This may cause some computational problems when determining the

[25] Remember that a smaller Fiedler value indicates a less intensive communication between the nodes.

eigenvectors of P. Simple methods for finding second dominating eigenvector are discussed in Appendix B.3.3.

Verma and Meilă introduced in [349] an additional criterion preventing further partitioning of too small subsets. Thereby, only clusters of "reasonable" size are returned by their algorithm.

Algorithm 5.3 Algorithm SM-VM for the determination of minimal normalized cuts, [349]

```
1: Compute the similarity matrix $S$; its entries $s_{ij}$ can be calculated by using the
   equation (5.1). Compute the degree matrix $D = \mathrm{diag}(S{\cdot}\mathbf{e})$.
2: Let $\mu_2$ denotes the second dominating eigenvalue of he matrix $P = D^{-1}S$ and let
   $\mathbf{y}_2$ be the corresponding eigenvector.
3: Let $\mathcal{V}$ denotes a list of subsets of the set $V$, which are potential candidates for
   further analysis. Analogously, $\mathcal{M}$ and $\mathcal{Y}$ are the lists containing the eigenvalues
   and eigenvectors computed for the subsets from $\mathcal{V}$. Initially $\mathcal{V} = \{V\}$, $\mathcal{M} = \{\mu_2\}$,
   $\mathcal{Y} = \{\mathbf{y}_2\}$. The lists are ordered in such a way that the first element in $\mathcal{V}$ is
   characterized by the minimal eigenvalue $\mu_2$.
4: $j = 1$
5: while $j < k$ do
6:    Let $A$ be the first element in $\mathcal{V}$, and let $v_2 \in \mathcal{M}$, $\mathbf{u}_2 \in \mathcal{Y}$ denote the eigenvalue
      and the eigenvector corresponding to this set.
7:    Partition the set $A$ into two subset $A_1, A_2$ applying a rounding strategy to the
      vector $\mathbf{u}_2$.
8:    For each subset $A_i$ compute the eigenpair $(\mu_{i,2}, \mathbf{y}_{i,2})$, $i = 1, 2$.
9:    Modify the list, i.e. $\mathcal{V} = \mathcal{V}\backslash\{A\} \cup \{A_1, A_2\}$, $\mathcal{M} = \mathcal{M}\backslash\{v_2\} \cup \{\mu_{1,2}, \mu_{2,2}\}$, $\mathcal{Y} =$
      $\mathcal{Y}\backslash\{\mathbf{u}_2\} \cup \{\mathbf{y}_{1,2}, \mathbf{y}_{2,2}\}$.
10: end while
11: return partition of the set $V$ into $k$ disjoint groups.
```

It has been noted in [311], [253] and [349] that if the entries of the Fiedler vector $\mathbf{w}$ are of the form

$$w_j = \begin{cases} \alpha \text{ if } v_j \in C \\ \beta \text{ if } v_j \in \overline{C} \end{cases} \tag{5.63}$$

then the pair $(C, \overline{C})$ is optimal normalized cut, and $Ncut(C, \overline{C}) = \lambda_2$, where λ_2 is the Fiedler value. The eigenvector satisfying the equation (5.63) is said to be piecewise linear, or constant vector – see Fig. 5.10 on page 184.

If $k > 2$, we can alternatively construct the matrix W, whose columns are the dominating eigenvectors $\mathbf{w}_2, \ldots, \mathbf{w}_k$. Treating the rows of this matrix as new coordinates of the objects (nodes), we use a clustering algorithm to partition the set V, [252].

5.2.5.3 NJW – spectral algorithm of Ng, Jordan & Weiss

When analysing the SM algorithm, [311], Ng, Jordan and Weiss applied in [269] the second representation of the generalized eigenproblem, i.e they used the equation (5.57) from Section 5.2.4. In this manner they obtained the algorithm described in pseudocode 5.4. Two features distinguish this new procedure from the SM algorithm:

(a) Instead of the Laplacian $\mathfrak{L}$ its complement $\mathfrak{L}^d = \mathbb{I} - \mathfrak{L}$ is used. As a consequence, the eigenvectors corresponding to the dominating (largest) eigenvectors of $\mathfrak{L}^d$ are used. According to the authors, this simplifies and clarifies theoretical analysis without changing the entries of these vectors. But it is not a cosmetic difference only. Computing the dominating eigenpairs is easier than computing eigenpairs with lowest eigenvectors – see the methods mentioned in Appendix B.3.3.1.
(b) More important difference concerns the way, in which the final results are obtained. In many cases, particularly when S is a block-diagonal matrix, its dominating eigenvectors are almost parallel to straight lines with different slopes – see Fig. 5.7. Thus, the authors mentioned suggest that the spectral coordinates determined in Step 3 of the algorithm should be projected onto the unit k-dimensional unit sphere – see Step 4. These new points are clustered next by a fast clustering algorithm, like k-means, FCM or any of its variants described in Section 3.3.5. The k-means algorithm is initialized[26] according to the method (d) described in Sect. 3.1.3.

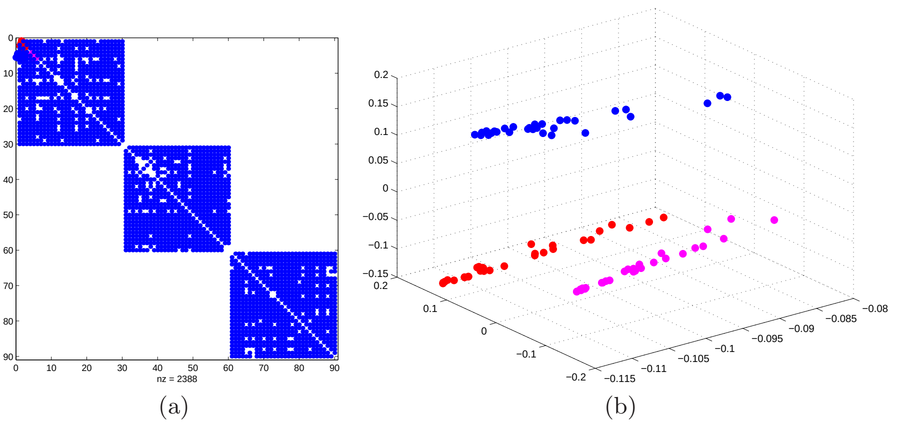


Fig. 5.7. Spectral representation of three clusters: (a) matrix S representing the data, (b): three dominating eigenvectors of the laplacian $\mathfrak{L}^d$.

Ng, Jordan and Weiss justify their algorithm as follows, [269]. Suppose that the dataset consists of $k = 3$ clusters C_1, C_2, C_3, and the objects are indexed in such a way that the first $|C_1|$ elements belong to the group C_1, the objects numbered $|C_1|+1, \ldots, |C_2|$ belong to the second group, and remaining objects

[26] Alternatively a method described by Fisher and Poland in [126] can be used.

Algorithm 5.4 NJW – the algorithm designed by Ng, Jordan and Weiss, [269]

```
1: Compute the similarity matrix $S$ using Gaussian kernel (5.1) and set $s_{ii} = 0$, $i = 1, \dots, m$.
2: Compute the degree matrix $D = \mathrm{diag}(S\cdot\mathbf{e})$ and the complement $\mathfrak{L}^d = D^{-1/2}SD^{-1/2}$.
3: Determine $k$ largest eigenvalues of the matrix $\mathfrak{L}^d$. The corresponding eigenvectors are the columns of the matrix $W \in \mathbb{R}^{m\times k}$
4: Create the matrix $U$ by normalizing rows of the matrix $W$, i.e.
       $$u_{ij} \leftarrow w_{ij} \cdot \Big(\sum_{t=1}^{k} w_{it}^2\Big)^{-1/2} \qquad (5.64)$$
5: Taking the rows of the matrix $U$ as $k$-dimensional coordinates divide the set of objects into $k$ disjoint groups.
6: return partition of the set $V$ into $k$ groups.
```

belong to the third cluster. Let $\widehat{S}$ be a block diagonal similarity matrix such that $\widehat{s}_{ij} > 0$, if the objects i and j belong to the same group, and $\widehat{s}_{ij} = 0$, if they belong to different groups. Such a matrix can be obtained when the groups are sufficiently far away from each other. In this case the Laplacian $\mathfrak{L}^d$ is block-diagonal, i.e.

$$\widehat{S} = \begin{bmatrix} \widehat{S}^{(1)} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \widehat{S}^{(2)} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \widehat{S}^{(3)} \end{bmatrix}, \ \mathfrak{L}^d = \begin{bmatrix} \mathfrak{L}^{d(1)} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathfrak{L}^{d(2)} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathfrak{L}^{d(3)} \end{bmatrix}$$

where $\mathfrak{L}^{d(i)} = [\widehat{D}^{(i)}]^{-1/2}\widehat{S}^{(i)}[\widehat{D}^{(i)}]^{-1/2}$, $\widehat{S}^{(i)}$ is a positive submatrix (block) of size $|C_i| \times |C_i|$, and $\widehat{D}^{(i)}$ is the degree matrix of i-th block.

The spectrum of the matrix $\mathfrak{L}^d$ is the union of the spectra of the blocks $\mathfrak{L}^{d(i)}$. Hence, the $k = 3$ dominating eigenvalues of the matrix $\mathfrak{L}^d$ are the dominating eigenvectors of each block $\lambda_1^{(i)}$, $i = 1, \dots, k$. Let $\mathbf{w}^{(i)}$ denote the dominating eigenvector of the block $\mathfrak{L}^{d(i)}$, $i = 1, \dots, k$. The columns of the matrix

$$W = \begin{bmatrix} \mathbf{w}^{(1)} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{w}^{(2)} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{w}^{(3)} \end{bmatrix} \qquad (5.65)$$

are eigenvectors of the Laplacian $\mathfrak{L}^d$. Moreover, if $R \in \mathbb{R}^{k\times k}$ is an orthogonal matrix (i.e. $RR^{\mathsf{T}} = R^{\mathsf{T}}R = \mathbb{I}$), the columns of the matrix $V = WR$ are also eigenvectors of the Laplacian $\mathfrak{L}^d$. Hence, the matrix W computed in step 3 of the NJW algorithm, can be written down as follows

$$W = \begin{bmatrix} W^{(1)} \\ W^{(2)} \\ W^{(3)} \end{bmatrix} = \begin{bmatrix} \mathbf{e}_{|C_1|} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{e}_{|C_2|} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{e}_{|C_3|} \end{bmatrix} R$$

where $W^{(i)} \in \mathbb{R}^{|C_i| \times k}$, and $\mathbf{e}_{|C_i|}$ stands for the unit vector of length $|C_i|$, $i = 1, \dots, k$. As a consequence we obtain the following lemma:

Lemma 5.2.2 *Let S be a block-diagonal matrix consisting of k positive blocks $S^{(1)}, \dots, S^{(k)}$. Then there exists k orthogonal vectors $\mathbf{r}_1, \dots, \mathbf{r}_k$ such that the rows $\mathbf{w}_j^{(i)} \in W^{(i)}$ of matrix W satisfy the condition*

$$\mathbf{w}_j^{(i)} = \mathbf{r}_i, \; j = 1, \dots, m, \; i = 1, \dots, k \tag{5.66}$$

□

In other words, there are k orthogonal vectors $\mathbf{r}_1, \dots, \mathbf{r}_k$, located on k-dimensional unit sphere. The entries of these vectors determine spectral coordinates of the data points – see step 4 of the NJW algorithm. Figure 5.8 illustrates this property.

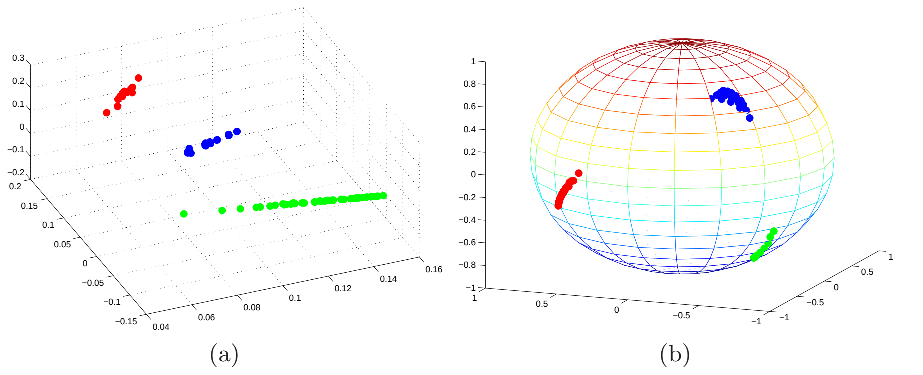


Fig. 5.8. NJW algorithm applied to the `data3_2` dataset: (a) Spectral coordinates of the data, and (b) their projection onto the unit sphere

This property remains true if some regularity constraints are satisfied. Namely:

(a) $\lambda_{k+1} < 1$,
(b) there are no outliers in the clusters, and
(c) the clusters are compact, i.e. they cannot be split into meaningful subclusters.

More rigorous description of these constraints, together with a justification for their introduction, can be found in [269].

It is interesting that the similar observations with respect to the adjacency matrix derived from a similarity matrix were forwarded by Wu *et al.*, [367]. These authors noted that the nodes projected onto the adjacency eigenspace exhibit an orthogonal line pattern and nodes from the same cluster are situated along the same line. This phenomenon was explained by referring to the graph perturbation theory. As a result, the authors propose an even simpler clustering algorithm.

5.2.5.4 DaSpec algorithm

The algorithms already described are based on a silent assumption that the dominating eigenvectors of similarity matrix S (or a Laplacian derived from S) contain the information allowing for the partition of the data. But this assumption is not always true, particularly when the clusters are of different size and shape. In such situations the eigenvectors suggested by these algorithms can provide redundant information which results in wrong partitions.

In the previous section we noted, after the paper [269], that if a similarity matrix is block-diagonal, then the optimal partition can be extracted from k dominating eigenvectors of the matrix $D^{-1/2}SD^{-1/2}$. This statement remains valid as long as the spectral gap $\lambda_k - \lambda_{k+1}$ is sufficiently large. The property was analyzed in depth by Rebagliati and Verri in [292]. These authors suggest that when the spectral gap is rather low, then $m_1 > k$ eigenvectors must be computed. Here, m_1 is a parameter chosen by a user in such a way that *extended* spectral gap, $\lambda_k - \lambda_{m_1+1}$ is sufficiently large. The final partition is obtained from a k-dimensional subspace of the space spanned by these m_1 eigenvectors. Other methods oriented towards selecting important eigenvectors are described in [193, Sect. 3.3].

An interesting solution to this problem was proposed by Shi, Belkin and Yu in [312], where the algorithm `DaSpec` (*Data Spectroscopic Clustering*) is described[27]. The number of chosen eigenvectors estimates the (unknown) number of clusters in the data. This number depends on the value of the parameter ω, and the authors suggest to choose this value as

$$\omega = \frac{95\% \text{ quantile } \{q_1, \dots, q_m\}}{\sqrt{95\% \text{ quantile } \chi^2_n}} \tag{5.67}$$

where q_i denotes 5% quantile of the set of the values $\{\|\mathbf{x}_i - \mathbf{x}_j\|, j = 1, \dots, m\}$, and χ^2_n is the χ^2 distribution with n degrees of freedom. Such a choice allows for selecting ω as the smallest value that covers most data points (i.e. 95% of them) having a certain number (5% of sample size) of neighbors within the "range" of the kernel. Further, the authors state, [312, p. 3973], that

> (...) this particular choice of ω works well in low-dimensional case. For high-dimensional data generated from a lower-dimensional structure, such as an n-manifold, the procedure usually leads to an ω that is too small. We suggest starting with ω defined in (5.67) and trying some neighboring values to see if the results are improved, maybe based on

[27] A number of interesting remarks associating this approach with quantum mechanics can be found in Martin's note "Data Spectroscopy: Gaussian Kernels and Harmonic Oscillators" available at `https://charlesmartin14.wordpress.com/2012/10/`.

some labeled data, expert opinions, data visualization or trade-off of the between and within cluster distances.

The reasoning used in [312] generalizes the argumentation presented in Sect. 5.2.5.3. When the clusters are sufficiently separable, i.e. the distance between any pair of the centroids is sufficiently large, then the representation (5.65) can be applied. The authors of [312] assume that the dataset comes from a mixture

$$P = \sum_{j=1}^{k} \pi_j P_j$$

where P_j is the probability distribution according to which the elements of j-th cluster were generated. If $\pi_j \approx 1/k$, and the sample is sufficiently large, the clusters are well balanced. In such a situation cluster separability depends on the parameters of the distributions P_j only, cf. Property 1 in [312]. On the other hand, if some weights π_j are significantly different from the other ones, then the dominating eigenvalues of the blocks representing "small" groups are much lower than the eigenvalues of the remaining blocks. Consequently, these small eigenvalues are outside the set of k dominating eigenvectors of the similarity matrix. But a characteristic feature of the dominating eigenvector of a single block is that all its elements have the same sign. Because of some disturbances, caused by the remaining components of the mixture, we assume that the elements of an eigenvector $\mathbf{w} = (w_1, \ldots, w_m)^{\mathsf{T}}$ are approximately (with precision ϵ) of fixed sign if

$$\min_{1 \le l \le m} w_l > -\epsilon, \text{ or } \max_{1 \le l \le m} w_l < \epsilon \tag{5.68}$$

where $\epsilon = \big(\max_{1 \le l \le m} |w_l|\big)/m$.

The eigenvectors that have no sign changes up to precision ϵ can be treated as a generalization of the representation (5.65). In this situation we do not need to use any clustering algorithm. If $\mathbf{w}^{(1)}, \ldots, \mathbf{w}^{(k)}$ are the vectors satisfying the above condition, we can use the rule

$$(\mathbf{x}_i \in C_j) \Leftrightarrow j = \arg\max_{1 \le l \le k} w_i^{(l)} \tag{5.69}$$

In summary, `DaSpec` algorithm consists of few steps presented in pseudocode 5.5.

Such a reasoning can also be used in estimation of the parameters of the mixture (3.42), see [312] for details.

5.2.6 Maximization of group connectivity

Minimization of the indices (5.11) or (5.39) is only one possible approach used in spectral clustering. A cluster can also be viewed as a group of highly similar objects, or – in terms of the similarity graph representing the data – as the set of nodes with a high number of interconnections. Let

Algorithm 5.5 Data spectroscopic clustering (`DaSpec`) algorithm, [312]

```
1: input data: The data $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$, $\omega > 0$ and the thresholds $\epsilon_j > 0$.
2: Compute the elements of the similarity matrix, $(K_m)_{ij} = \frac{1}{m} \exp\left( - \frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\omega^2} \right)$.
3: Determine the eigenvalues $\lambda_1, \ldots, \lambda_{m_1}$ and eigenvectors $\mathbf{w}_1, \ldots, \mathbf{w}_{m_1}$, $j = 1, \ldots, m_1 < m$ of the matrix $K_m$.
4: Among the set of eigenvectors obtained in previous step identify those that have no sign changes up to precision $\epsilon_j$ (in terms of the definition (5.68)). Let $k$ be the number of the identified eigenvectors.
5: Assign the objects to appropriate groups by using the rule (5.69).
```

$$assoc(C) = \sum_{\mathbf{x}_i, \mathbf{x}_j \in C} s_{ij} \tag{5.70}$$

stand for the total weight of the connections among the objects from the set C. Now, we are interested in the partition $\{C, \overline{C}\}$, that maximizes the index

$$Massoc(C, \overline{C}) = assoc(C) + assoc(\overline{C}) \tag{5.71}$$

By analogy to the minimization problem (5.16), we are looking now for a matrix M satisfying the condition

$$\boldsymbol{\chi}^{\mathsf{T}} M \boldsymbol{\chi} = \sum_{(v_i, v_j) \in E} s_{ij} (\chi_i + \chi_j)^2 \tag{5.72}$$

Here, we sum the weights of the links joining the nodes belonging to the same group. A matrix M will satisfy the above condition if its elements are of the form[28]

$$m_{ij} = \begin{cases} \mathsf{d}_i \text{ if } i = j \\ s_{ij} \text{ if } i \neq j \text{ and } (i,j) \in E \\ 0 \text{ otherwise} \end{cases} \tag{5.73}$$

By analogy to the combinatorial Laplacian with the elements l_{ij}, defined in the equation (5.18), we will call the matrix M signless Laplacian of the graph G. Indeed, for any indices $i, j = 1, \ldots, m$ there holds $m_{ij} = |l_{ij}|$. Many properties of the matrix M are studied in [85].

The matrix M can be written as

$$M = D + S = 2D - L \tag{5.74}$$

and

$$\begin{gathered} \max \frac{\boldsymbol{\chi}^{\mathsf{T}} M \boldsymbol{\chi}}{\boldsymbol{\chi}^{\mathsf{T}} \boldsymbol{\chi}} \\ \text{s.t. } \chi_i = \pm 1, \ i = 1, \ldots, m, \ \boldsymbol{\chi}^{\mathsf{T}} \boldsymbol{\chi} = m \end{gathered} \tag{5.75}$$

[28] We assume thet G is a simple graph, i.e. $s_{ii} = 0$ for all $i = 1, \ldots, m$.

is a counterpart to the problem (5.20).

Liu [237] proves that all the entries of the eigenvector corresponding to the maximal eigenvalue of the matrix M, are positive. This results from the following facts: (a) all eigenvalues of $M \in \mathbb{R}^{m \times m}$ are real numbers since M is symmetric, and (b) since G is a connected graph, then M is an irreducible matrix, i.e. by Perron-Frobenius theorem there exists exactly one eigenpair $(\lambda_{max}, \mathbf{w}_1)$, such that λ_{max} is the dominating eigenvalue and all the elements of $\mathbf{w}$ have the same sign. Thus, the solution to the problem (5.75) is the eigenvector corresponding to the second maximal eigenvalue of the matrix M.

Let λ_2 be the Fiedler value of the Laplacian L, and let μ_2 denote the second maximal eigenvalue of the signless Laplacian $M = 2D - L$. Then, see [237, Sect. 3.3])

$$\frac{\lambda_2}{\mu_2} \approx \frac{cut(C, \overline{C})}{Massoc(C, \overline{C})} \tag{5.76}$$

Although the corresponding eigenvectors of the matrices L and M induce similar partitions, the above observation offers additional information about the quality of final partition.

Consider again the generalized eigenproblem (5.45). By substituting $L = 2D - M$ we can state that if its solution is the pair $(\lambda, \mathbf{w})$, then

$$(2D - M)\mathbf{w} = \lambda D\mathbf{w} \Rightarrow M\mathbf{w} = (2 - \lambda)D\mathbf{w} \Rightarrow M\mathbf{w} = \mu D\mathbf{w} \tag{5.77}$$

where $\mu = 2 - \lambda$. Thus, if the pair $(\lambda, \mathbf{w})$ solves the generalized eigenproblem (5.45), then the pair $(2 - \lambda, \mathbf{w})$ is the solution to the generalized eigenproblem $M\mathbf{w} = \mu D\mathbf{w}$, [96, Sect. 3.2].

Analogously to the normalized Laplacian, we can define normalized signless Laplacian

$$\mathfrak{M} = D^{-1/2} M D^{-1/2} = \mathbb{I} + D^{-1/2} S D^{-1/2} \tag{5.78}$$

Let $(\lambda, \mathbf{w})$ be an eigenpair of the normalized Laplacian $\mathfrak{L} = \mathbb{I} - D^{-1/2} S D^{-1/2}$. As $(\mathbb{I} - D^{-1/2} S D^{-1/2})\mathbf{w} = \lambda\mathbf{w}$, it follows that $(1 - \lambda, \mathbf{w})$ is an eigenpair of the matrix $D^{-1/2} S D^{-1/2}$. Similarly, if $(\mu, \mathbf{w})$ is an eigenpair of the normalized signless Laplacian $\mathfrak{M}$, then the eigenpair of the matrix $D^{-1/2} S D^{-1/2}$ has the form $(\mu - 1, \mathbf{w})$. In other words, knowing the eigenpair $(\alpha, \mathbf{w})$ of the matrix $D^{-1/2} S D^{-1/2}$ we can determine the eigenpair $(1 - \alpha, \mathbf{w})$ of normalized Laplacian and the eigenpair $(1 + \alpha, \mathbf{w})$ of the normalized signless Laplacian.

In particular the following theorem is of interest

Theorem 5.2.4 *[237] The cost of minimal normalized (balanced) cut can be computed by solving the eigenproblem*

$$\mathfrak{M}\mathbf{w} = \mu\mathbf{w} \tag{5.79}$$

If $\mathbf{w}_2$ is the eigenvector corresponding to the second maximal eigenvalue μ_2 of the matrix M, then the vector $D^{-1/2}\mathbf{w}_2$ is the solution to the problem of minimization of the normalized cut. □

5.2.7 Some examples

Let us start with `2rings` dataset. The similarity matrix, representing this set, is shown in Fig. 5.9. We thresholded the values of this matrix setting $s_{ij} = 0$ if $s_{ij} < 0.02$. As we see from the left panel, upon setting $\omega = 0.3$ we obtain almost block diagonal matrix. In this situation, the Fiedler vector of the Laqplacian $\mathfrak{L}$ is piecewise linear – see Fig. 5.10(a). Similarly, the points with the coordinates determined by the entries of two dominating eigenvectors of the matrix $\mathfrak{L}^d$ lie on the straight lines each of which corresponds to appropriate group – Fig. 5.11(a). Projection of these points onto the unit sphere (circle in this case) are shown in Fig. 5.12(a).

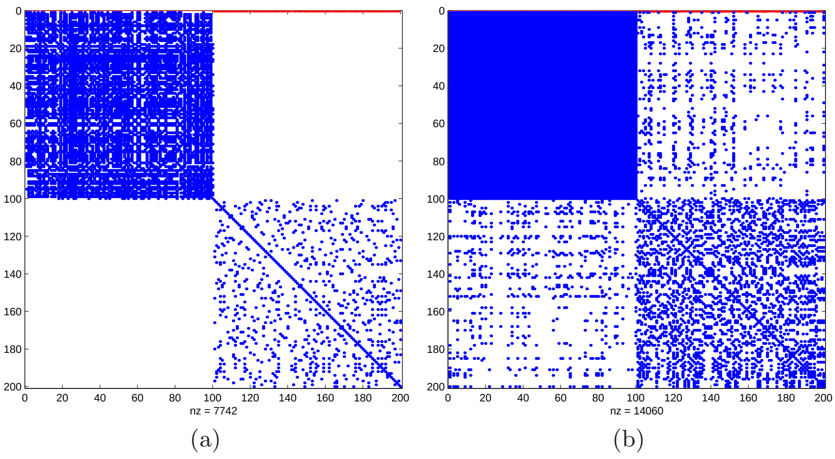


Fig. 5.9. Similarity matrix representing the dataset `2rings` for two different values of the ω parameter: (a) $\omega = 0.3$, (b) $\omega = 1$.

When the parameter ω increases from 0.3 to 1.0, the block diagonal structure of matrix S becomes less apparent. The Fiedler vector, computed for the new version of the Laplacian $\mathfrak{L}$ loses its previous linear nature, but still allows for the correct classification of the objects. The nonlinear fragment depicted in Fig. 5.10(b) corresponds to the less dense group of objects located in the outer ring. In effect, the points with the coordinates determined by two dominating eigenvectors of the matrix $\mathfrak{L}^d$ form a richer geometric structure, shown in Fig. 5.11 (b). The lower right corner is occupied by the patterns of objects from the dense cluster, while the cloud in the upper part of the figure represents the points from the outer ring. Fig. 5.12(b) shows projection of these points onto the unit sphere. Despite the loss of clear property, mentioned in section 5.2.5.3, it is still possible to reconstruct faithfully group membership.

It is important to note, that the `DaSpec` algorithm from the previous section deals pretty well with the subsets generated from a mixture of probability dis-

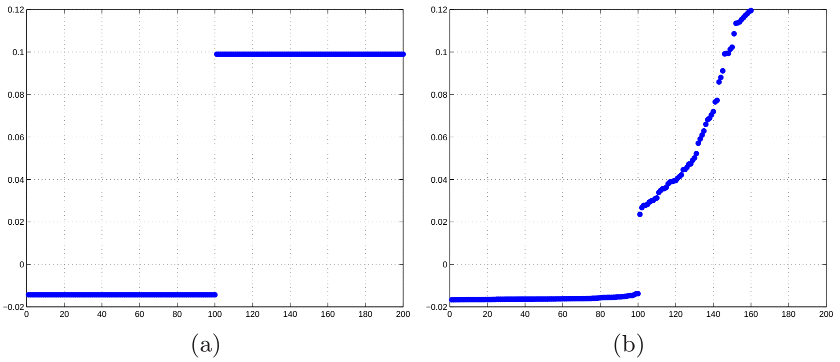


Fig. 5.10. Fiedler vector of the normalized Laplacian derived from the similarity matrix: (a) $\omega = 0.3$, (b) $\omega = 1$.

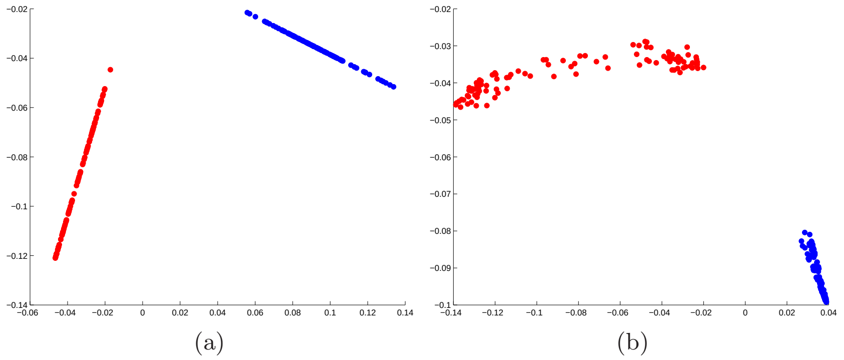


Fig. 5.11. The points with the coordinates determined by two dominating eigenvectors of the matrix $\mathfrak{L}^d$: (a) $\omega = 0.3$, (b) $\omega = 1$.

tributions, but it fails in the case of data displaying more complicated geometric structure.

Consider now the `iris` dataset. Taking $\omega = 0.35$ and using the `NJW` algorithm we obtain the results shown in Fig. 5.13. For $k = 3$ we obtain the 3-dimensional patterns of the 4-dimensional original set if objects – see left panel. Here the *Iris setosa* group is denoted by red points, *Iris Versicolor* – as blue dots, and *Iris Virginica* – as green dots. The resulting representation confirms the well known fact that the flowers from *Iris Setosa* species form a group well separated from two remaining groups. Projection of these 3-dimensional points onto the unit sphere is depicted in Fig. 5.13(b). Again, *Iris Setosa* flowers occupy a separated area of the sphere, while the remaining objects are arranged along a common meridian.

By applying the k-means algorithm to the points located on the unit sphere (see Step 5 of the `NJW` algorithm) we obtain the following confusion matrix

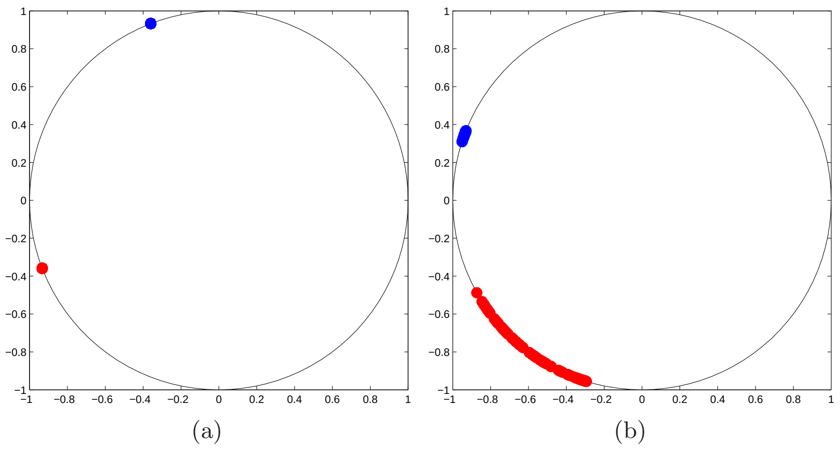


Fig. 5.12. Projection of the points from Figure 5.11 onto the unit sphere: (a) $\omega = 0.3$, (b) $\omega = 1$.

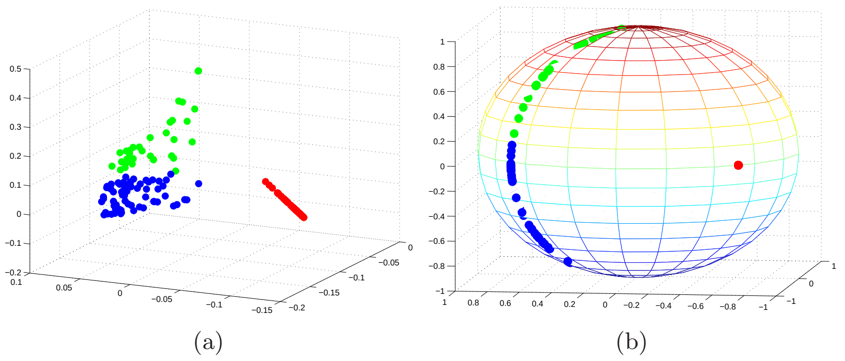


Fig. 5.13. Results obtained from the `NJW` algorithm applied to `iris` dataset: (a) spectral coordinates determined for the original data, and (b) their projection onto the unit sphere

Prediction	True classification		
	Setosa	*Vericolour*	*Virginica*
Setosa	50	0	0
Vericolour	0	50	0
Virginica	0	14	36

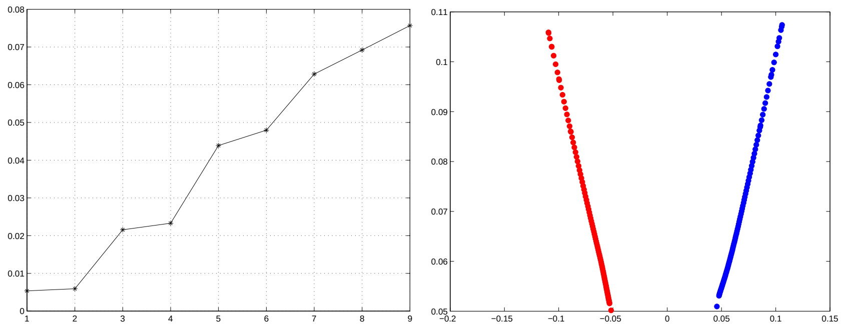
Hence the accuracy of this grouping is 90.67%. On the other hand, by running 100 times the k-means algorithm on the original data we obtain the average accuracy equal to 89.16%.

Various variants of the spectral clustering algorithm are discussed by von Luxburg in [351]. She suggest the following instantiation of the algorithm:

(a) When the similarity matrix is computed by using the equation (5.1), we obtain a densely connected graph. According to von Luxburg, at least at the initial stage of the analysis, we should construct the so-called t-nearest neighbor graph[29]. That is, a node v_i is connected with node v_j if v_j is among the t-nearest neighbors of v_i. The value of t should be of the order of $\ln(m)$, which ensures the connectivity of the resulting graph. Such a procedure is specially useful when the groups are of different density. Further, the resulting similarity matrix is a sparse matrix.

(b) The parameter ω used in the equation (5.1) should be approximately equal to the average t-distance. This is illustrated for the `2spirals` datasets. The average distance among the elements of this set, computed for different values of t, is depicted in Fig. 5.14(a). As the set consists of 190 objects, the best value of t is $t_{opt} = \lceil \ln(190) \rceil = 6$. This implies that $\omega \approx 0.05$. By choosing this value we obtain spectral coordinates depicted on Fig. 5.14(b). As we see, the groups represented in these coordinates are perfectly separated.

(c) Lastly, according to von Luxbrug, instead of symmetric Laplacian, (5.51) the nonsymmetric matrix $L = \mathbb{I} - D^{-1}S$ should be used.



(a) (b)

Fig. 5.14. Spectral analysis of the dataset `2spirals`: (a) The average values of the t-distance among the elements from this set computed for $t \in \{1, \dots, 9\}$. (b) Spectral representation of the original objects obtained for $\omega = 0.05$

5.2.8 Tuning the algorithm

Spectral clustering is successful if the similarity matrix is block-diagonal, [269]. In most cases this matrix is computed by using the Gaussian kernel (5.1) with the parameter $\gamma = 1/(2\omega^2)$. Unfortunately, there are no hints on how to choose the proper value of this parameter. Worse, with a fixed value of ω, the similarity between any pair of data points depends on their Euclidean distance mainly.

[29] In general such a graph is a directed graph, as the neighborhood relationship is not symmetric. See [351] for recipes of making this graph undirected.

This means that in case of data with more complicated structure, it is hard to model the data distribution adequately, what results in turn, in poor quality of spectral clustering. Even if it is possible to use a single value of ω, in many cases of real data its proper value must be chosen from a rather narrow range. Jia *et al.* provide in [193, Sect. 3.1] a review of various approaches to the construction of similarity measure.

In this section we discuss some simple methods allowing for adapting the algorithm to these requirements, that is: how to choose the proper value of ω and how to amplify the block-diagonal structure of the similarity matrix.

5.2.8.1 Choosing ω parameter

The effectiveness of spectral clustering algorithm heavily depends on the parameter ω, appearing in the equation (5.1). Ng, Jordan and Weiss proposed in [269] an iterative approach relying upon choosing as ω a value from an interval $[\omega_{min}, \omega_{max}]$, such that the dispersion of a resulting partition – measured in terms of the index (3.1) – is minimal. However, this approach requires multiple runs of the algorithm with different values of ω. Moreover, we must compute the eigenvectors of the Laplacian obtained for each value of ω.

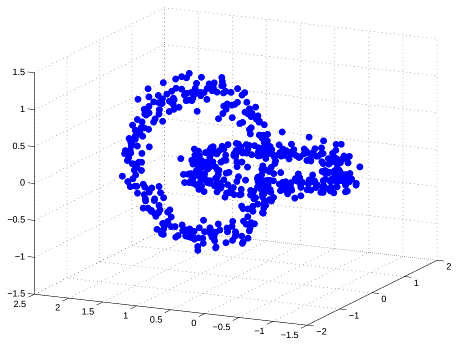
A slightly more efficient method of choosing proper value of ω has been proposed in [315]. It consists of four steps:

1. Compute, using the equation (5.1), the matrix $S(\omega)$ for several different values of the parameter ω.
2. For each value of ω compute the sum $\Sigma(\omega) = \sum_{i=1}^{m} \sum_{j=1}^{m} s_{ij}(\omega)$.
3. Plot $\Sigma(\omega)$ against the values of ω in doubly logarithmic scale. The plot should have two asymptotes: for $\omega \to 0$ and $\omega \to \infty$.
4. Choose the value ω^* from the middle of the linear part of the plot.

In case of the dataset `rings`[30], depicted in Fig. 5.15(a), we obtain the plot shown in Fig. 5.15(b). An approximated value of ω^* lying in the middle of the linear segment of the plot is $\omega^* \approx 0.2$.

Another, rather expensive, method relies upon drawing a histogram of distances between pairs of objects from the set X. The presence of cluster structure can be observed by noting local maxima on the plot – see Fig. 5.16. The first maximum corresponds to the average intra-cluster distance and the other ones to between-cluster distances. By choosing ω close to the first maximum, we assign high values of similarity to the pairs of objects located within a common cluster. Thus, the S matrix should have the block-diagonal structure. This method works very well for clearly separable clusters, as illustrated in Fig. 5.16(a). The plot represents pairwise distances between the objects from `data6_2` dataset. In case of data from figure 5.15(a), the histogram has a less clear structure. However,

[30] It is available from the Web page `http://www.dr-fischer.org/pub/blockamp/index.html` associated to the paper [126].


(a)

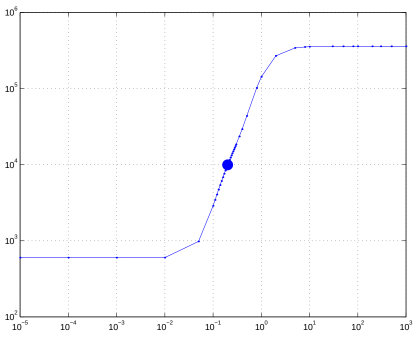
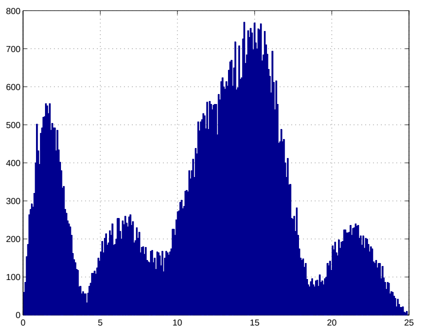

(b)

Fig. 5.15. (a): The set `rings` containing 600 points inside two interlocked rings. (b): Doubly logarithmic plot of the values of $\Sigma(\omega)$ values obtained for different values of ω. The blue dot denotes approximated value of the parameter ω.

even here one can see that the first maximum occurs in the vicinity of the point $\omega = 0.2$. A disadvantage of this method is that it cannot be automated.


(a)

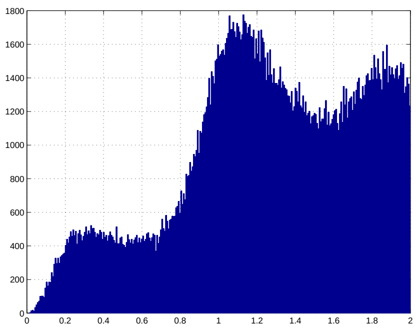

(b)

Fig. 5.16. Histogram of distances between the elements of two datasets: `data6_2`, and (b): `rings`

In the local (contextual) method suggested, e.g., in [382] an individual ω value is assigned to each datapoint. These authors assume that ω_i is equal to the distance between i-th datapoint and its t-th nearest neighbor; the suggested value for this t is $\lfloor \ln(m) \rfloor$. This corresponds to our earlier suggestion, mentioned in the point (b) in the previous section. In this case the equation (5.1) takes the form

$$s_{ij} = \exp\left(- \frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\omega_i \omega_j} \right) \tag{5.80}$$

Fischer and Poland suggest in [126] another method: ω_i is the value solving the equation

$$\sum_{j=1}^{m} \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\omega_i} \right) = \tau \tag{5.81}$$

where $\tau > 0$ is a "neighborhood size" parameter; its suggested value is $2n + 1$. According to these authors there should be two neighbors for each dimension, plus one because a point is a neighbor of itself.

Let $\widetilde{s}_{ij}$ stands for the similarity between a pair of objects. The similarity matrix with such values is asymmetric. It can be symmetrized using the transformation $s_{ij} = \min(\widetilde{s}_{ij}, \widetilde{s}_{ji})$. Such a procedure is especially useful when the clusters are of different densities. Another method is discussed in Section 5.2.5.4.

5.2.8.2 Amplifying the block-diagonal structure of a matrix

The choice of the "proper" value of ω is only one side of designing an efficient clustering algorithm. Consider again the `rings` data from Fig. 5.15(a). By taking $\omega = 0.2$ we obtain the similarity matrix shown in Fig. 5.17(a). It is a band matrix[31]. More precisely, it is a block-band matrix consisting of two band matrices each of which represents a unique cluster. Fisher and Poland propose in [126] a method of "expanding" the bands, i.e. the method of transforming a block-band matrix to a block-diagonal form shown in Fig. 5.17(b).

Although the empirical results show the correctness of the clustering resulting from the spectral decomposition of the block-band matrix, the structure of such a matrix is usually observable only in a very narrow range of small values of the parameter ω. For instance, in case of the `rings` data $\omega \in [0.13, 0.24]$.

Treating the graph as a resistive circuit, cf. Appendix D, the similarity between the nodes of the graph can be replaced by the conductivity (not to be confused with a specific conductance equation (5.35) on p.162), i.e. the inverse of resistance. Let, as before, $L = [l_{ij}]$ be a combinatorial Laplacian. Let us define the matrix $G = [g_{ij}]$ as follows:

$$g_{ij} = \begin{cases} 1 & \text{if } i = 1, j = 1 \\ 0 & \text{if } i = 1, j > 1 \\ l_{ij} & \text{if } i > 1 \end{cases} \tag{5.82}$$

Except for its first row, this matrix is identical to the Laplacian L derived from a similarity matrix S. However, while the Laplacian is singular, the matrix G is invertible. Thus, the approximate value of conductivity between nodes i and j is determined from the equation[32]

$$c_{ij} = \left(g_{ii}^{-1} + g_{jj}^{-1} - g_{ij}^{-1} - g_{ji}^{-1} \right)^{-1} \tag{5.83}$$

[31] $A \in \mathbb{R}^{m \times m}$ is called a band matrix if $a_{ij} = 0$ whenever $|i - j| > w$ for some positive integer w, called the bandwidth.

[32] See [126] for details and full justification of this equation.

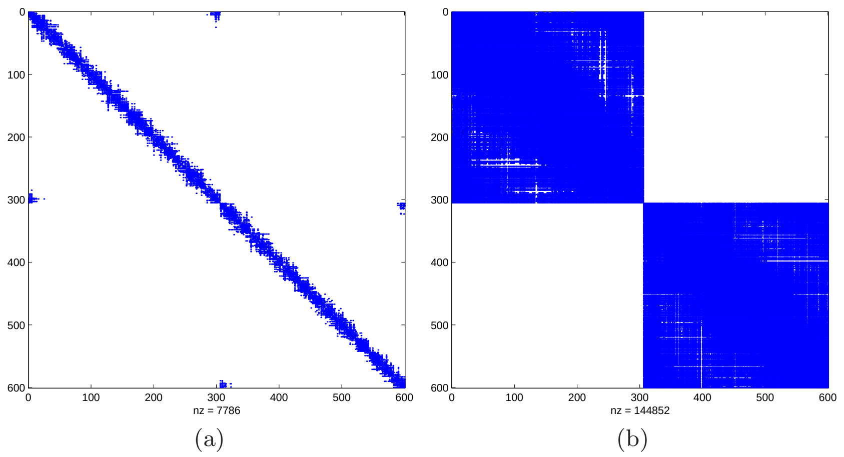


Fig. 5.17. (a): Block-band similarity matrix representing the data from Fig. 5.15(a) The first blok consists of the band assigned to the data points numbered from 1 to 300. The remaining data points are represented by the second band. (b): Block-diagonal conductivity matrix C corresponding to the matrix S

Although the matrix C with the elements as above was derived from the original Laplacian its block-diagonal nature is substantially amplified. This phenomenon is illustrated in Fig. 5.17(b), where the conductance matrix, corresponding to the similarity matrix from Fig. 5.17(a), has been shown.

5.3 Random walks on graphs

Spectral clustering has few important disadvantages. First, there is no precise guidance on how to calculate the similarity between pairs of objects. Even the simple recipe (5.1) requires careful choice of the parameter ω. Further, similarity measures ignore almost completely the geometrical structure of the entire dataset. Other disadvantages are pointed out by Nadler and Galun in [263]. They note that the algorithms used for spectral clustering typically translate the local information (expressed by the numbers s_{ij} describing the strength of relationship between pairs of objects) into global information (expressed by the eigenvectors) used as the basis for grouping. Using only local information does not allow, these authors write, to treat the normalized cost $NCut$ as a reliable measure of the quality of partition. Moreover, even if a proper measure of similarity is used, the first few eigenvectors of an appropriate matrix are not sufficient for the adequate separation of groups in situations, in which these groups differ in size, density and volume[33].

Hence, we need another perspective. The generalized eigenproblem $L\mathbf{y} = \lambda D\mathbf{y}$, introduced in Section 5.2.3, is equivalent to the problem $D^{-1}L\mathbf{y} = \lambda\mathbf{y}$.

[33] But see Section 5.2.5.4 for at least a partial remedy.

But $D^{-1}L = D^{-1}(D - S) = \mathbb{I} - D^{-1}S$, where

$$\overline{P} = D^{-1}S \tag{5.84}$$

is a row-stochastic matrix[34], which can be treated as a transition matrix describing random walk on a graph associated to the similarity matrix S. Moreover, if $(\lambda, \mathbf{v})$ is an eigenpair of the matrix $\overline{P}$, then $(1-\lambda, \mathbf{v})$ is an eigenpair corresponding to the matrix $D^{-1}L$, i.e. it is a solution to the generalized eigenproblem. Thus, a strictly technical modification of the Laplacian (used up to this moment as a main vehicle for clustering) moves the problem of clustering into the domain of random walk. The parameters of this walk (see Section D.1.2), governed by a reversible Markov chain with transition matrix (5.84), allow to assign new meanings to the notion of cluster. In this section we discuss subsequently these meanings. A review of various similarity measures, derived from the characteristics of the random walk can be found, e.g., in [130] and [291]. An important feature of this new approach is the ability to take into account a variety of contextual information, what is important e.g. in recommender systems [59], [130].

5.3.1 Random walk on undirected graph

The vast majority of the results obtained in this area concerns the undirected graphs. The basic notions of random walk on graphs are given in Appendix D.

5.3.1.1 Simple interpretations

The row-stochastic matrix $\overline{P}$ from equation (5.84) characterizes a Markov chain. Its elements $\overline{p}_{ij}$ represent the probability of moving from node i to node j. Roughly speaking, the larger the transition probability between two nodes, the greater the possibility that these two nodes belong to the same cluster. Below, a more rigorous formalization of this observation is given.

The stationary (or equilibrium) distribution $\boldsymbol{\pi}$ of the Markov chain determined by the similarity matrix S has the elements

$$\pi_i = \frac{\mathsf{d}_i}{\sum_{j=1}^{m} \mathsf{d}_j} = \frac{\mathsf{d}_i}{\operatorname{vol} V} \tag{5.85}$$

This distribution admits two simple and important properties:

(a) $\boldsymbol{\pi}^{\mathsf{T}}\overline{P} = \boldsymbol{\pi}^{\mathsf{T}}$,
(b) $\pi_i p_{ij} = \pi_j p_{ji}$.

First property states that the distribution $\boldsymbol{\pi}$ does not change after one step (and in consequence: after any number of steps). The second property states that the random walk corresponds to a reversible Markov chain.

[34] i.e. $\sum_{j=1}^{m} \overline{p}_{ij} = 1$ for each row $i = 1, \dots, m$.

Let P_{AB} denote the probability of moving, in one step, from a state $i \in A \subset V$ to any state in the set B. Since the walker starts from a state $i \in A$, the initial probability distribution has the form

$$p_A(i) = \begin{cases} \frac{\mathsf{d}_i}{\sum_{j \in A} \mathsf{d}_j} & \text{if } i \in A \\ 0 & \text{otherwise} \end{cases}$$

and the requested probability equals to[35]

$$P_{AB} = \sum_{j \in B} \sum_{i \in A} \frac{\mathsf{d}_i p_{ij}}{\operatorname{vol} A} = \frac{\sum_{i \in A, j \in B} s_{ij}}{\operatorname{vol} A} = \frac{cut(A,B)}{\operatorname{vol} A} \tag{5.86}$$

As a consequence, we obtain the following lemma:

Lemma 5.3.1 *Let G be be an undirected and connected graph. Let $\{C, \overline{C}\}$ be a partition of the set of nodes, and let $P_{C\overline{C}}$ denotes the probability of leaving the set of states C (and moving to the set $\overline{C}$) defined in equation (5.86). Then:*

(i) $NCut(C, \overline{C}) = P_{C\overline{C}} + P_{\overline{C}C}$
(b) If G is an unweighted graph, i.e. $S = A$ and $volC \leq \frac{1}{2}volV$, then

$$P_{C\overline{C}} = \Phi(C) \tag{5.87}$$

where $\Phi(C)$ denotes the conductance[36] of the set C. □

This lemma offers a probabilistic interpretation of the quantities used to measure the quality of a partition. Part (i) of this Lemma has been proved in [252]. It states that if $\{C, \overline{C}\}$ is a partition with low normalized cut, then the two probabilities $P_{C\overline{C}}$ and $P_{\overline{C}C}$ are also small. In other words, if the walker goes to a state belonging to cluster C or $\overline{C}$, then the probability of leaving this cluster is small. To be more illustrative, imagine that the undirected graph is a city map: the links of the graph represents streets and the nodes of this graph correspond to the streets intersections. If the walker went to the quarter covered by many local streets and only a few of them lead to other districts[37], he will spend a lot of time there before he gets out of there.

The second part of the Lemma, used e.g. in [323], provides a probabilistic interpretation for the conductance of a "small" set (i.e. a set with small volume).

Let us note, that minimization of normalized cut must not lead to the result obtained by minimization of conductance. This fact is illustrated in Fig. 5.18, reproduced from [335]. Here, $C_1 = \{1, 2, 3\}$ is the cluster yielding the

[35] Here we use the fact that $p_{ij} = s_{ij}/\mathsf{d}_i$, and in the last equation we make use of the definition (5.11).

[36] We introduced this notion in Section 5.2.2, see eqn. (5.35). See also Appendix C.2.1.1.

[37] An excellent example of this situation is the old town (*medina*) of Fez in Morocco. There is almost nine thousand streets!

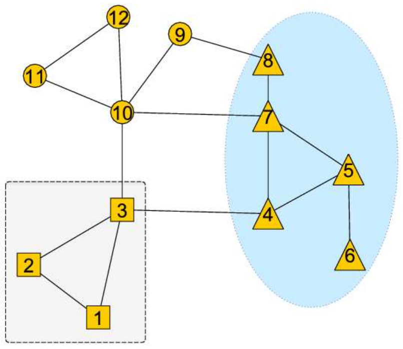


Fig. 5.18. The cluster corresponding to low normalized cut (squares) vs. the one corresponding to low conductance (triangles).

minimal normalized cut, while $C_2 = \{4, \dots, 8\}$) is the set with minimal conductance. One can check that $Ncut(C_1, \overline{C}_1) = 1/3$ and $\Phi(C_2) = 3/13$. The conductance $\Phi(C_1) = 1/4$, is slightly greater that $\Phi(C_2)$. And the normalized cut $Ncut(C_2, \overline{C}_2) = (3/13 + 3/19)$ exceeds by nearly 17% the cost $Ncut(C_1, \overline{C}_1)$.

The next lemma, presented in [252], characterizes another important and interesting property of the spectral grouping algorithm.

Lemma 5.3.2 *Let $\overline{P} = D^{-1}S$ be a row stochastic matrix of size m and let $\Delta = \{C_1, \dots, C_k\}$ be a partition of the set of indices $I = \{1, \dots, m\}$. The matrix $\overline{P}$ has k piecewise constant (different from zero) eigenvectors if and only if the sums*

$$\sigma_{is} = \sum_{j \in C_s} \overline{p}_{ij} \tag{5.88}$$

are fixed for each $i \in C_s$, and the matrix R with elements

$$r_{ss'} = \sum_{i \in C_s} \sigma_{is'} = \sum_{i \in C_s} \sum_{j \in C_{s'}} \overline{p}_{ij} \tag{5.89}$$

is nonsingular. □

The matrix $\overline{P}$, satisfying conditions of this Lemma, is said to be a block-stochastic matrix. This lemma is an elegant generalization of the results presented in [311] and [195]. The authors of these papers pointed out that if G consists of k connected components, then minimization of the normalized cuts leads to correct results. As can be seen, this condition can be generalized. Further, this lemma establishes a connection to the so-called *lumpability*[38], what allows for the efficient reduction of the size of states of the Markov chain. This idea is yet extended by E, Li and Vanden-Eijnden [116], as well as by Lafon and Lee [224].

[38] See e.g. J.G. Kemeny, J.L. Snell, *Finite Markov Chains* (Second ed.). Springer-Verlag, 1976.

But the lumpability is only one side of the coin. It is important that the eigenvalues of the matrix R corresponding to piecewise linear eigenvectors must be greater than the remaining m-k eigenvalues of the original matrix $\overline{P}$. Meilă and Shi [252] call these remaining eigenvalues *spurious* eigenvalues. If these spurious eigenvalues are much smaller than the first k eigenvalues – we can automatically determine the number of clusters[39]. Unfortunately, it is hard to define in advance the gap between the "real" and the spurious eigenvalues. We can only say that [252]:

(a) If the matrix R is close to the unit matrix, then its eigenvalues are close to 1.
(b) If the rows of the matrix $\overline{P}$ are sufficiently mutually similar, then the spurious eigenvalues are close to zero.

In fact, the situation is not so clear. In Fig. 5.19, the distribution of eigenvalues of the normalized Laplacian $D^{-1/2}AD^{-1/2}$, computed for the sets `2rings` and `2spirals` are depicted (these values are sorted in ascending order). As we see, it is hard to distinguish between the subsequent values. Further, the eigenvalues usually depend on the ω parameter, and this, in turn, significantly affects the quality of the induced partition.

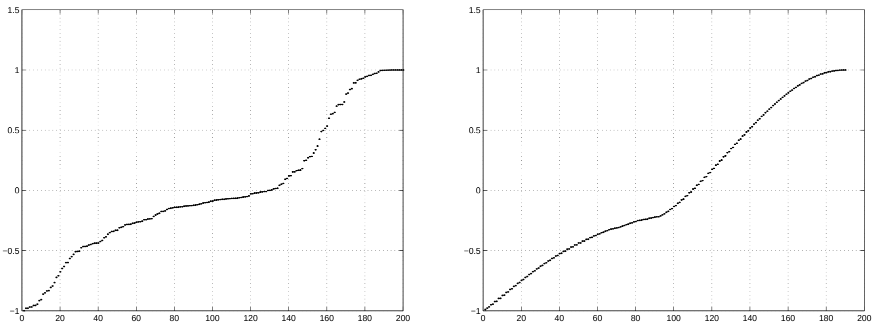


Fig. 5.19. Eigenvalues of the symmetric Laplacian computed for the sets `2rings` and `2spirals`. Left panel corresponds to the set `2rings` and value $\sigma^2 = 1$, while the right – to the set `2spirals` and the value $\sigma^2 = 0.007$.

5.3.1.2 Clustering according to the nodes potential

We shall now refer to the metaphor, based on the theory of electric networks. By a resistive circuit we understand a connected and undirected graph $G = (V, E)$, in which we assign to each edge $\{u, v\}$ a resistance $R_{uv} > 0$; equivalently, the conductance of an edge $\{u, v\}$ is $C_{uv} = 1/R_{uv}$. If V_u denotes the voltage at node u, then according to Ohm's Law the current I_{uv} that flows from u to v is equal to

[39] A similar idea was suggested by Shi, Belkin and Yu in [312] – see Sect. 5.2.5.4.

$$I_{uv} = \frac{V_u - V_v}{R_{uv}} = (V_u - V_v)C_{uv}$$

By Kirchhoff's First Law, the total current flowing out of any node is equal to the sum of currents flowing into this node. If $N(u)$ stands for the set of neighbors of the node u un graph G, then this law can be written as

$$\sum_{v\in N(u)} I_v = \sum_{v\in N(u)} (V_v - V_u)C_{vu}$$

or, equivalently

$$V_u = \sum_{v\in N(u)} p_{uv}V_v \tag{5.90}$$

where

$$p_{uv} = \frac{C_{uv}}{\sum_{w\in N(u)} C_{uw}} \tag{5.91}$$

Assuming that $p_{uv} = 0$ if $\{u, v\} \notin E$, we build a stochastic matrix with the elements p_{uv}. This shows, that the resistive circuit is equivalent to a Markov chain, described by the transition matrix P, or to the random walk on the graph G with the probability p_{uv} of moving from node u to node v. A nice description of this idea can be found in [110].

To force the current flow in such a network we must connect the battery to two nodes. Let us number the nodes of the graph in such a way that the battery poles are connected to the nodes numbered 1 and 2, and assume that $V_1 = 1, V_2 = 0$. Under such a setting we obtain the system of equations

$$\begin{aligned} V_1 &= 1\\ V_2 &= 0\\ V_i &= \sum_{j=1}^{m} p_{ij}V_j \end{aligned} \tag{5.92}$$

By denoting

$$\mathbf{V} = \begin{bmatrix} V_3\\ \vdots\\ V_m \end{bmatrix},\ T = \begin{bmatrix} p_{33} & \dots & p_{3m}\\ \vdots & & \vdots\\ p_{m3} & \dots & p_{mm} \end{bmatrix},\ \mathbf{C} = \begin{bmatrix} p_{31}\\ \vdots\\ p_{m1} \end{bmatrix} \tag{5.93}$$

we rewrite this system in the matrix form

$$\mathbf{V} = T\mathbf{V} + C$$

The unique solution to this equation is

$$\mathbf{V} = (\mathbb{I} - T)^{-1}\mathbf{C} \tag{5.94}$$

A reader should note that $(\mathbb{I}-T)^{-1}$ is nothing else but the so-called fundamental matrix of an absorbing Markov chain, see e.g. [188].

The earlier equation can be written in another, equivalent, form by using the notations introduced in Section 5.1. Remembering that $p_{ij} = s_{ij}/\mathsf{d}_i$ and noting that $p_{ii} = 0$ for any $i = 1, \ldots, m$, we obtain

$$\widetilde{L} = \mathbb{I} - T = \begin{bmatrix} \mathsf{d}_3 & -s_{34} & \ldots & -s_{3m} \\ -s_{43} & \mathsf{d}_4 & \ldots & -s_{4m} \\ \vdots & & \vdots & \\ -s_{m3} & -s_{m4} & \ldots & \mathsf{d}_m \end{bmatrix}, \; \widetilde{C} = \begin{bmatrix} s_{31} \\ \vdots \\ s_{m1} \end{bmatrix} \tag{5.95}$$

The matrix $\widetilde{L}$ is the Laplacian of the graph, from which the rows and columns corresponding to the absorbing states, i.e. the states numbered 1 and 2, are deleted. Similarly, $\widetilde{\mathbf{C}}$ is the weight vector describing connections between the node numbered 1 and the remaining nodes, numbered $3, 4, \ldots, m$. Under such a setting the Kirchhoff equation takes the form $\widetilde{L}\mathbf{V} = \widetilde{\mathbf{C}}$. Thus

$$\mathbf{V} = \widetilde{L}^{-1}\widetilde{\mathbf{C}} \tag{5.96}$$

Wu and Huberman noted in [364] that – since the connections among the nodes belonging to the same cluster are more intense than the connections among the nodes belonging to different clusters – the voltages assigned to the nodes from a single cluster are close to each other. This way we obtain new definition of cluster: it is a set of nodes with similar voltages. The only problem is how to attach properly the battery to the circuit, since the poles of the battery must be attached to nodes from separated clusters.

Although the time needed to solve the system (5.96) is of the order $O(m^3)$, Wu and Huberman propose a method of the order $O(m+\mathfrak{m})$. We present it in the pseudocode 5.6. The summing operation from line 4 refers to the last equation in (5.92), which has the form

$$V_i = \frac{1}{\mathsf{d}_i} \sum_{j \in N(i)} s_{ij} V_j$$

This algorithm radically simplifies when G is an unweighted graph, since then $s_{ij} = 1$ for all $j \in N(i)$. Then

$$V_i = \frac{1}{\mathsf{d}_i} \sum_{j \in N(i)} V_j$$

Figure 5.20 illustrates the effect of the algorithm applied to the `karate` social network. The poles of the battery are attached to the nodes numbered 12 and 30. To partition the values of V_j, $j \in \{1, \ldots, 34\} \backslash \{12, 30\}$, the k-means algorithm with $k = 2$ was used.

Algorithm 5.6 An algorithm for solving the system of equations (5.96)

```
1: Initialization. $k = 0$, $V_1^{(0)} = 1$, $V_j^{(0)} = 0$, $j = 2, \dots, m$. Compute nodes degree $\mathsf{d}_i$,
   $i = 1, \dots, m$
2: for $k = 1$ to $k_{max}$ do
3:    for $i = 3$ to $m$ do
4:       $V_i^{(k)} = \frac{1}{\mathsf{d}_i} \sum_{j \in N(i)} s_{ij} V_j^{(k-1)}$
5:    end for
6: end for
```

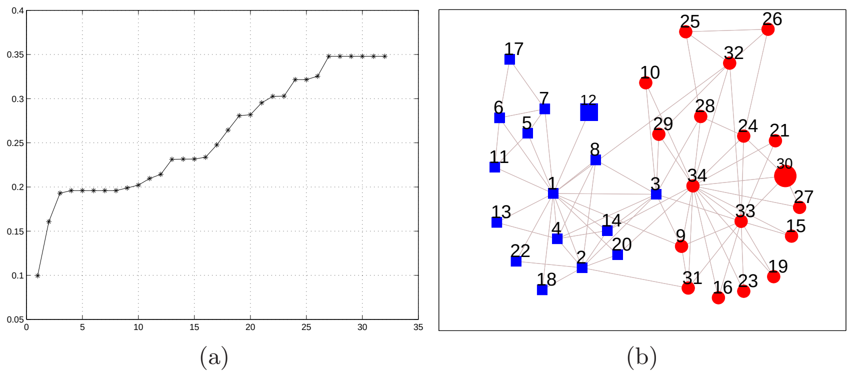


Fig. 5.20. Clustering the nodes of `karate` network by using Wu and Huberman algorithm [364]. (a): Voltages of the nodes in ascending order. For better visualization the values 0 and 1 are removed. (b): Partition of the nodes obtained by k-means algorithm applied to these voltages. The poles of the battery were attached to the nodes marked by large markers.

5.3.1.3 Resistance distance

One of important parameters, describing random walk, is the so-called hitting time h_{ij} defined as the expected number of steps before, starting from node i, the walker will visit the node j. The expected time for the random walk to travel from node i to node j and then back is called the commute time, $c_{ij} = h_{ij} + h_{ji}$. Various methods allowing for the efficient computation of these parameters are discussed in Appendix D.1.2.2, while the application of these notions in clustering is presented – for instance – in the papers [130] and [288].

It is important to note that the commute time corresponds to the so-called resistance distance r_{ij}, [68]. This last notion refers to the efficient resistance[40] between nodes i and j of the graph representing electric circuit. More precisely,

$$c_{ij} = 2\mathfrak{m} r_{ij}$$

This means that both c_{ij} and $\sqrt{c_{ij}}$ are distances. It should be noted that the commute distance c_{ij} decreases when the number of paths joining the nodes i

[40] Consult Section 1.3.4 of [110] for precise definition of efficient resistance.

and j increases and when the length of these paths decreases. This is an important property that distinguishes resistance distance from geodesic distance[41]. Knowing the matrix $C = [c_{ij}]$ we can design a counterpart of the k-means algorithm.

A careful analysis of the properties of the resistance distance was carried out by von Luxburg, Radl and Hein in [353]. These authors proved that when the size of the graph increases, the resistance distance loses its discriminative properties, since for large graphs $c_{ij} \approx 1/\mathsf{d}_i + 1/\mathsf{d}_j$. A review of different heuristics meant to rectify this defect is given in Section 4 of [353]. Another simple and quite efficient solution was proposed by Brand in [59] who proposed a cosine similarity measure[42].

An interesting application of the resistance distance in the analysis of empirical graphs is described in [59] and [130]. In these papers a method of constructing the graph G, representing a relational database is proposed. This can be of use in designing recommender systems and in analysis of text documents.

5.3.1.4 Grouping according to absorption time

Grouping according to the nodes potential, discussed in Sect. 5.3.1.2, is a simple method, but it has one important drawback: we must attach the battery to the right nodes. An interesting and more effective method relies upon treating a node s as the absorbing state of the Markov chain, corresponding to a given graph. Next, we assign to its neighborhood all the nodes that are absorbed by this state in shortest time. This method is an example of local clustering, discussed later. Yet we mention it here, as it uses one more parameter characterizing random walk on undirected graph.

Let P be a transition matrix, characterizing random walk on the graph G, and let s stands for the node, to which we want to assign its natural neighbors. According to the convention mentioned above, we treat the node s as the absorbing state. To simplify the notation, we re-number the nodes of the graph in such a way that the node s is number 1. Next, the matrix P is replaced by the matrix $\widehat{P}$ with the elements

$$\widetilde{p}_{ij} = \begin{cases} 1 & \text{if } i = j = 1 \\ 0 & \text{if } i = 1,\ j = 2, \dots, m \\ p_{ij} & \text{for } i = 2, \dots, m,\ j = 1, \dots, n \end{cases} \tag{5.97}$$

In other words, this matrix has the canonical form

$$\widetilde{P} = \begin{bmatrix} 1 & \mathbf{0} \\ \mathbf{r} & Q \end{bmatrix} \tag{5.98}$$

where $\mathbf{r} = (p_{21} \dots p_{m1})^{\mathsf{T}}$, Q is the matrix obtained by removing first row and first column from the matrix P , and $\mathbf{0}$ is the row vector consisting of $m-1$ zeros.

[41] i.e. the shortest path joining the nodes i and j.

[42] Recall that a similar measure is used in information retrieval to quantify similarity between a pair of documents.

The average number of steps, before the walker starting from any node $i \in \{2, \ldots, n\}$ will end up in the state s, equals to the appropriate element of the vector [43]

$$\mathbf{t} = (\mathbb{I} - Q)^{-1}\mathbf{e} = \sum_{k=0}^{\infty} Q^k \mathbf{e} \tag{5.99}$$

Direct application of this formula is expensive for large graphs. An approximate algorithm for determining the components of the vector $\mathbf{t}$ is described in the paper [272]. We sketch below the details of this method.

Let us refer to the spectral decomposition of the matrix Q^k

$$Q^k = \sum_{i=1}^{m-1} \lambda_i^k \mathbf{v}_i \mathbf{w}_i^{\mathtt{T}}$$

where λ_i is i-th eigenvalue of the matrix Q, and $\mathbf{v}_i$ (resp. $\mathbf{w}_i$) is the corresponding right (resp. left) eigenvector of this matrix.
Next, let

$$\widetilde{Q} = \widetilde{D}^{1/2} Q \widetilde{D}^{-1/2}$$

be a symmetric matrix similar to the matrix Q, and $\widetilde{D}$ be the generalized degree matrix from which s-th row and column were deleted. The matrices Q and $\widetilde{Q}$ have identical eigenvalues and the left and right eigenvectors of matrix Q can be computed from the eigenvectors $\mathbf{u}_i$ of matrix $\widetilde{Q}$, cf. Lemma B.3.3 on p. 235, as follows

$$\mathbf{v}_i = \widetilde{D}^{-1/2}\mathbf{u}_i, \ \mathbf{w}_i = \widetilde{D}^{1/2}\mathbf{u}_i$$

Using these observations we rearrange the equation (5.99) to the form

$$\begin{aligned} \mathbf{t} &= \sum_{k=0}^{\infty} Q^k \mathbf{e} \\ &= \mathbf{e} + \sum_{k=1}^{\infty} \sum_{i=1}^{n-1} \lambda_i^k \mathbf{v}_i \mathbf{w}_i^{\mathtt{T}} \mathbf{e} \\ &= \mathbf{e} + \sum_{k=1}^{\infty} \Big(\sum_{i=1}^{n-1} \lambda_i^k c_i \mathbf{v}_i \Big) \end{aligned} \tag{5.100}$$

where $c_i = \sum_j w_{ij} = \sum_j \sqrt{d_j} u_{ij}$.

If the spectral gap is sufficiently large, that is, the quotients $|\lambda_i/\lambda_1|$ are small numbers, the above equation can be approximated by the following expression

[43] The matrix $N = \mathbb{I} - Q$ is so-called fundamental matrix of an absorbing Markov chain. Its element n_{ij} is the expected number of times that the walker will be in state s_j before absorption when it starts in s_i. More details concerning this problem can be found e.g. in [110] or [188].

$$
\begin{aligned}
\widehat{\mathbf{t}} &= \mathbf{e} + \sum_{k=1}^{\infty} \lambda_1^k \left(c_1 \mathbf{v}_1 + \sum_{i=2}^{n-1} \left(c_i \frac{\lambda_i}{\lambda_1} \right)^k \mathbf{v}_i \right) \\
&\approx \mathbf{e} + c_1 \mathbf{v}_1 \sum_{k=1}^{\infty} \lambda_1^k \qquad (5.101) \\
&= \mathbf{e} + c_1 \frac{\lambda_1}{1-\lambda_1} \mathbf{v}_1
\end{aligned}
$$

The authors of the paper [272] noted that the vectors $\mathbf{t}$ and $\widehat{\mathbf{t}}$ are highly correlated even if the spectral gap is close to 1. Thus, we can use $\widehat{\mathbf{t}}$ instead of the original vector $\mathbf{t}$. This phenomenon is illustrated on Fig. 5.21. Here we computed, for the `karate` network, the mean-square between the difference of the vectors $\widehat{\mathbf{t}}$ and $\mathbf{t}$. It was assumed that the walker starts from subsequent nodes – Fig. 5.21(a). On panel (b) the specific values of the vector $\widehat{\mathbf{t}}$ are compared with the theoretical values; it was assumed there that the walker starts from the node 34. Note high correlation between these values.

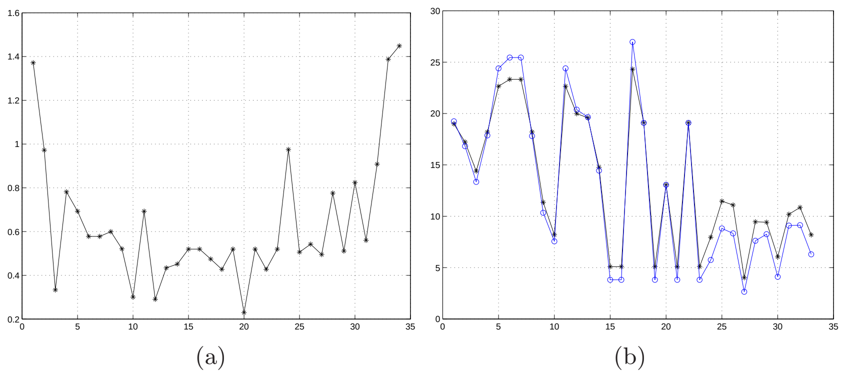


Fig. 5.21. Agreement between theoretical and approximate values of the absorption times. (a): Mean-square error obtained for the cases when the walker starts his walk from different nodes in `karate` network. (b) The absorption times computed for 33 nodes for the case when the walker starts from the node #32; theoretical values are denoted by $\circ$, and approximated values by $*$.

The vector $\mathbf{v}_1$ from the equation (5.101) is referred to as the local Fiedler vector, or Dirichlet-Fiedler vector. It is the eigenvector corresponding to the maximal eigenvalue of the matrix Q, obtained by removing from $\widetilde{P}$ the rows and columns, corresponding to the absorbing states. Equivalently, it is the vector of the form $\mathbf{v}_1 = \widetilde{D}^{-1/2}\mathbf{u}_1$, where $\mathbf{u}_1$ is the *dominating* eigenvector of the symmetric matrix $\widetilde{Q} = \widetilde{D}^{1/2} Q \widetilde{D}^{-1/2}$, and $\widetilde{D}$ is the degree matrix spanned over the set of non-absorbing states. The eigenvalues of $\widetilde{Q}$ are called Dirichlet eigenvalues. They satisfy the condition, see [75, Sect. 8.4], $-1 < \lambda_i \leq 1$. More observations on the role of Dirichlet eigenvalues and eigenvectors can be found in [78]

5.3.2 Application of the random walk idea: the `MCL` algorithm

The `MCL`, ***Markov CLustering*** algorithm, was proposed by van Dongen, who described the main idea of the algorithm in his Ph. D. thesis [343]. A short description of the algorithm can be found in [345], and its application in bioinformatics is presented in [118].

The readers interested in source code (in C++ and Pearl) can find many interesting remarks on author's webpage `http://micans.org/mcl/`. Other implementations, in Java and MATLAB, are available from `http://www.arbylon.net/projects/`. We should also mention the mailing list `http://listserver.ebi.ac.uk/pipermail/mcl-users/`, where the users of the program exchange their suggestions and observations.

Since the algorithm is extremely simple and scalable, it has found many applications in bioinformatics [118], [303], social network analysis [301], linguistics [109], and so on.

5.3.2.1 Basic version of the algorithm

The essence of the `MCL` algorithm is the following observation: if a random walker is placed in a node located in a dense area of a sparse graph G then she/he is able to leave that area after visiting the majority of nodes from this area (see Lemma 5.3.1 from p. 192). Obviously, the random walking idea as such, does not allow for extracting the clusters. After having walked sufficiently long, the walker will visit all the nodes of a (connected) graph. Thus, in the `MCL` algorithm, after few steps, the most important (probable) paths are identified, and later on the walker considers only these paths. The entire algorithm consists of three steps:

(i) We add the unit matrix to the adjacency matrix; such an operation corresponds to adding self-loops to each of the nodes. Next, we normalize each column of the resulting matrix. In this way a *column* stochastic matrix M is obtained. The element m_{ij} of this matrix corresponds to the probability of moving from the state (node) j to i in one time step.

(ii) Next, the walker makes k steps, which results in replacing the matrix M by the new matrix M^k; here k is a small integer, typically $k = 2$. The replacement $M \leftarrow M^k$ is called *expansion*.

(iii) The second important operator is that of *inflation*, Γ_r, where $r > 0$ is the so-called inflation coefficient; usually $r \approx 1.5 - 2$. The inflation relies upon: (a) powering each element of the matrix M, and (b) normalization of the resulting column of the matrix M, i.e.

$$(\Gamma_r M)_{ij} = \frac{m_{ij}^r}{\sum_{i=1}^n m_{ij}^r} \tag{5.102}$$

Here $(\Gamma_r M)_{ij}$ stands for the element belonging to the i-th row and j-th column of the matrix $\Gamma_r M$. In effect, the matrix $\Gamma_r M$ is also column-stochastic.

(iv) The steps (ii) and (iii) are repeated until M becomes idempotent, i.e. $M = M \cdot M$.

It is important that the above procedure converges very fast (after $10 - 20$ repetitions) to the idempotent matrix M. The resulting matrix is sparse: only few of its rows contain non-zero elements; these elements are used to reconstruct clusters in the graph. Granularity of the clusters depends on the parameter r. For small values of r we obtain a small number of large clusters, while for large values of r – a much larger number of smaller clusters is obtained. More of technical details can be found in [343, Sect. 10.4].

The convergence of the algorithm can be improved by introducing "pruning" of the small probabilities. That is, if $m_{ij}^r < \epsilon$, where ϵ is a small number, then we set $m_{ij}^r = 0$. Such a method works well if the diameters of the clusters are rather small. Other variants of pruning are discussed in [343, p. 136].

Below, we summarize the `MCL` algorithm in the form of the pseudocode 5.7.

Algorithm 5.7 `MCL` algorithm

```
Require: $A$ – adjacency matrix, $r > 0$ – inflation coefficient
1: $A \leftarrow A + I$        // add self-loops
2: $M \leftarrow AD^{-1}$      // prepare column-stochastic matrix
3: while ($M$ is not idempotent) do
4:     $M \leftarrow$ expand($M$)
5:     $M \leftarrow$ inflate($M, r$)
6:     $M \leftarrow$ prune($M$)
7: end while
8: return Matrix $M$
```

The functions `expand`, `inflate` and `prune` mentioned in lines 4-6 of the pseudocode realize the corresponding operations, described before.

The matrix M, constructed in step 2 of the algorithm, defines a stochastic flow. Each column $\mathbf{m}_j$ describes the outflow of the probability from the j-th node, and the j-th row defines inflow to the node j. Thus, the *MCL* process can be considered in terms of expansion and reduction of the flow. During expansion, flows from a given node to other nodes occur: these nodes are attainable in various ways from the given node. Multiple repetitions of expansion only result in that the columns of matrix M will tend to the dominant eigenvector of matrix $M = (M + \mathbb{I})D^{-1}$. The introduction of inflation prevents this process by increasing the flow between nodes belonging to a common cluster and reducing the flow between nodes from different clusters. Finally, at the end of the iterations, nodes that belong to a single cluster will direct the flow to one of the nodes, called *attractor* of this cluster. If v_j is such an attractor, the elements of j-th row occupying positions corresponding to the nodes from this cluster are close to 1. An important property of the algorithm is that it does not require prior knowledge about the number of clusters.

5.3.2.2 Disadvantages of the algorithm

The MCL algorithm – as all other algorithms – is not perfect. In case of large graphs the iterations become time consuming, and even worse – many small clusters are produced.

As noted by Satuluri and Parthasarathy in [301], at the initial iterations the matrix M is rather dense and the expansion requires $O(\sum_{j=1}^{m} \mathsf{d}_j^2)$ operations (recall that m is the number of nodes in the graph). In later iterations, when pruning comes into play, this number decreases to $O(m\eta^2)$, where η stands for the average number of non-zero elements in the columns of M. At the initial iterations, η is of order 10^3, which entails difficulties. Figure 5.22 illustrates the changes in the density of matrix M in subsequent steps (here the network karate was used).

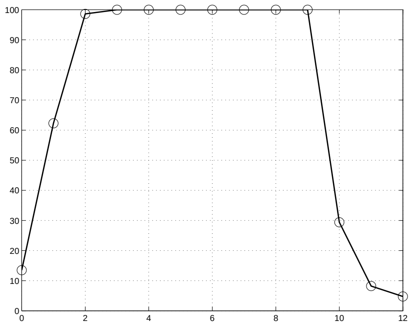


Fig. 5.22. The density of matrix M (expressed in percent) of the karate network in subsequent iterations of the MCL algorithm. Initially the matrix has 156 non-zero elements, i.e. 13.4948% of its total elements. Matrix density grows rapidly to 100%, and after 9 iterations it drops sharply

Satuluri and Parthasarathy note another disadvantage, called results fragmentation. For instance, in the case of a protein-protein network consisting of 4,741 nodes, they obtained 1416 clusters! Varying the inflation coefficient r only slightly improved the granulation of clusters.

To overcome these disadvantages, Satuluri presents in [301], [303], and [302] some modifications. In the first two papers a "compactification" is proposed. It relies upon "gluing" together nodes of the graph, applying MCL to the reduced graph, and next applying the inverse transformation to get the final results. A similar approach has been used in the GRACLUS, [101], and METIS, [199], algorithms. In the third paper, [302], a sparsification of the entire graph was proposed; it is performed by identifying and removing redundant edges. A reader interested in details is referred to the original papers.

Another feature of the MCL algorithm was observed and posted by Lars Juhl Jensen[44]. Namely, the algorithm may produce clusters, containing the nodes, which do not communicate with each other in a given graph. To illustrate this phenomenon, he constructed a small graph depicted in Fig. 5.23. The algorithm generates (for a wide range of values of the inflation coefficient) five clusters including the so-called unnatural cluster $\{X, Y\}$. Suprisingly, if we remove some the nodes, e.g. J and K, correct clustering is obtained.

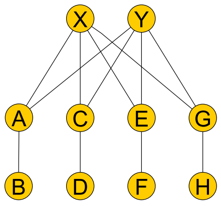


Fig. 5.23. Unnatural clusters produced by the MCL algorithm. If the inflation coefficient $r \in [1.734, 3.418]$, the algorithm returns five clusters: $\{A, B\}$, $\{C, D\}$, $\{E, F\}$, $\{G, H\}$ and $\{X, Y\}$. The last cluster is factitious (unnatural).

A serious consequence of this phenomenon is described in the paper published in *Nature*[45] in which an analysis of a protein-protein network consisting of 2,708 nodes was reported. Using the MCL algorithm, the authors extracted 547 clusters (protein complexes). Jensen and his Ph.D. student observed that 9 clusters among these obtained represent disjoint subgraphs[46]. It seems that Dongen noted this problem: his program is equipped with an option allowing to delete such clusters[47].

5.4 Local methods

The algorithms discussed in this chapter can be used for extraction of clusters in datasets as well as for cutting graphs into appropriate number of subgraphs. In this last case, apart from high computational complexity, the algorithms in question have at least two disadvantages, [226]. First, the spectral mapping not always produces the best cut. Second, in the case of large empirical graphs, unbalanced clusters are frequently returned. For this reason, especially in the case

[44] See "Analysis: Markov clustering and the case of the unnatural clusters", `http://larsjuhljensen.wordpress.com/2010/03/01/`.

[45] . N.J. Krogan *et al.* Global landscape of protein complexes in the yeast Saccharomyces cerevisiae. *Nature*, Vol 440—30 March 2006—DOI:10.1038/nature04670.

[46] See "Analysis: Markov clustering and the case of the unsupported protein complexes", `http://larsjuhljensen.wordpress.com/2010/03/03/`

[47] See his post `http://listserver.ebi.ac.uk/pipermail/mcl-users/2010-January/000072.html`.

of large graphs, a different approach has been proposed: starting with a small but consistent set of nodes, called *seed*, we search for the "natural neighbors" forming a "community" (or a collection of such communities), [16]. Here, by the community we understand a set of nodes $C \subset V$ of low conductance (computed as in the equation (5.35)). All the nodes forming such a community communicate with the seed (or its subset) also included in this community. Low conductance means that only a small fraction of edges with one end in the community have the second end outside this community. In other words, the nodes belonging to the community C communicate with each other more often than with nodes outside C. A review of alternative methods of community identification can be found e.g. in [129], [141], [267] and [268]. Interestingly, the low value of the conductance $\Phi(C)$ does not imply high intensity of connections between the nodes from C. Hence, the local methods can be used, first of all in graph clustering, but we can also try to apply such methods in the search for a set of objects sufficiently similar to a given object $\mathfrak{x}_i \in \mathfrak{X}$.

The first formal solution to the problem defined above was given by Spielnan and Teng in [321] and [323]. Their algorithm, called `Nibble`, allows to identify a set C together with the node $v \in V$ in time proportional to the size of this set.

The algorithms allowing to extract local communities are important for at least two reasons. First, they appear to be very effective, as they search only the "interesting" part of the graph (and the remaining subgraph, spanned over the nodes from the set $\overline{C}$ is ignored). Second, these algorithms allow for identifying small subsets existing inside very large graphs. Andersen, Lang and Chung in a series of papers [13]-[16] proposed an interesting modification of `Nibble`, called `PageRank-Nibble`.

One of characteristic features of empirical graphs is small average distance between any two nodes[48], which means a rapid increase (with distance) in the number of neighbors of the nodes forming a seed. This does not preclude the existence of communities that communicate with members of other communities only through a small number of connections (and therefore the cost of cutting such a community is small). It is usually assumed that the nodes forming a seed represents a unique community. Lang and Andersen proved in [16], that if there exists in a graph G, for a given seed, a well defined community, then their algorithm allows for extracting it. Let us repeat once again: the advantage of the local approach is that the time complexity of executing these procedures is proportional to the size of the generated clusters, and not – as in the case in conventional methods – to the size of the input graph, [14].

Below we present only a brief outline of the local clustering method; an interested reader is referred to the literature cited above. Important applications of this approach in bioinformatics are presented in the paper [241]. It is worth noting that the approach outlined here is not the only possible way to determine the natural neighborhood of some nodes of an empirical network. Methods developed in the context of social networks provide alternative solutions.

[48] See e.g. Watts, D.J.; Strogatz, S.H. Collective dynamics of 'small-world' networks. *Nature* 393(6684), 1998, 409-10. DOI:10.1038/30918

Remark 5.4.1 *The following considerations apply to unweighted graphs. Thus we will use in this section the adjacency matrix A instead of the similarity matrix S. The off-diagonal elements of A take the values of 0 or 1.* □

5.4.1 The `Nibble` algorithm

The essence of the algorithm is to propose an order in which a walker will visit the nodes of an undirected graph $G = (V, E)$. At a first glance, it would seem that we should consider the nodes closest to the starting node $u^* \in V$. But most of empirical graphs, like social networks, or protein-protein networks, have small diameter[49], and such a solution becomes ineffective. Spielman and Tang proposed in [323] to order the nodes according to the probability of visiting the nodes during a short random walk starting from the node u^*.

Let A be the adjacency matrix of the graph G. If the graph contains self-loops, then $a_{ii} = z_i$ if there are z_i such loops in node i. Let D be the degree matrix and let

$$\widehat{P} = (AD^{-1} + \mathbb{I})/2 \tag{5.103}$$

be the *column* stochastic matrix, representing *lazy* random walk. In other words, if there are no self-loops and a random walker is in node u, then in the next time steps she/he will stay, with the probability 1/2, in this node or with the probability $1/(2 \cdot |N(u)|)$ she/he will move to a neighbor node from the set $N(u)$. Such a construction of the matrix P guarantees that the random walk has the stationary distribution $\boldsymbol{\pi}$ (consult Section 5.3.1.1). Moreover, P is diagonally dominant, i.e. $\widehat{p}_{ii} \geq \sum_{j \neq i} |\widehat{p}_{ij}|$ for each row i.

The `Nibble` algorithm refers to the property mentioned in Lemma 5.3.1 on p. 192, which states that if S is a set with small conductance, then the probability of leaving it by the random walker is also small. Careful analysis, presented in [323], leads to a simple algorithm, which can be summarized as follows:

(a) Suppose that a random walk starts from a node $u^* \in V$. This implies the following starting probability distribution, defined on the set of states (nodes of the graph)

$$\mathbf{p}^{(0)}(v) = \chi_{u^*}(v) = \begin{cases} 1 \text{ if } v = u^* \\ 0 \text{ otherwise} \end{cases} \tag{5.104}$$

(b) If, after the k-th step of random walk, the probability distribution over the set of states is $\mathbf{p}^{(k)}$, then in the next step it has the form $\mathbf{p}^{(k+1)} = \widehat{P}\mathbf{p}^{(k)}$.

(c) To improve the convergence of the algorithm, Spielman and Teng introduced censored walk, in which the original probability vector $\mathbf{p}^{(k+1)}$ is approximated by

[49] A reader is referred to the Appendix C where basic notions from graph theory are summarized.

$$[p^{(k+1)}]_\epsilon(v) = \begin{cases} p^{(k+1)}(v) & \text{if } p^{(k+1)}(v) \geq \epsilon \mathsf{d}_v \\ 0 & \text{otherwise} \end{cases} \tag{5.105}$$

where ϵ is a small number.
From the formal standpoint, $[\mathbf{p}^{(k+1)}]_\epsilon$ is no longer a stochastic vector, but it was proved in [323], that such an approximation provides a correct cut, as well (and, what is important: reduces the time of the computations).

(d) For a given vector $\mathbf{p}$ we define the ordering function $q\colon V \to \mathbb{R}$ of the form

$$q(v) = \frac{p_v}{\mathsf{d}_v} \tag{5.106}$$

Let o stands for the ordering of the nodes $\{1, \dots, m\}$ such that

$$q(v_{o(1)}) \geq q(v_{o(2)}) \geq \cdots \geq q(v_{o(m)})$$

and let $S_j^{\mathbf{p}}$ denotes the set of nodes of the form

$$S_j^{\mathbf{p}} = \{v_{o(1)}, \dots, v_{o(j)}\} \tag{5.107}$$

Algorithm 5.8 `Nibble` – algorithm for local identification of clusters, [323]

```
1: Input data: Undirected graph $G = (V, E)$, starting node $u^*$, maximum value of
   conductance, $\phi$, integer $b$.
2: declare maximal number of steps, $k_{max}$ and compute the value $\epsilon$.
3: set $r_0 = \chi_{u^*}$
4: for $k = 1, \dots, k_{max}$ do
5:     $p^{(k)} = \widehat{P} r^{(k-1)}$
6:     $r^{(k)} = [p^{(k)}]_\epsilon$
7:     if $(\exists j \in V)\colon \Big(\Phi(S_j^{p^{(k)}}) \leq \phi\Big) \wedge \Big(2^b \leq \mathrm{vol}\,(S_j^{p^{(k)}}) \leq \frac{5}{6}\mathrm{vol}\,(V)\Big)$ then
8:         return the set $C = S_j^{p^{(k)}}$ and STOP
9:     end if
10: end for
11: return $C = \emptyset$
```

The essence of the algorithm is to determine the sequence of vectors

$$\begin{aligned} \mathbf{p}^{(k)}) &= \begin{cases} \boldsymbol{\chi}_{u^*} & \text{if } k = 0 \\ P\mathbf{r}^{(k-1)} & \text{otherwise} \end{cases} \\ \mathbf{r}^{(k)} &= [\mathbf{p}^{(k)}]_\epsilon \end{aligned} \tag{5.108}$$

while for every $k \in \{1, \dots, k_{max}\}$ we check whether among the collection of the sets $S_j^{\mathbf{p}^{(k)}}$, there is one for which the conductance $\Phi(S_j^{\mathbf{p}^{(k)}})$ does not exceed the value ϕ, and its volume is not too small and not too big[50].

[50] Spielman i Teng add one more condition, but, to save simplicity of presentation, we omit it.

The `Nibble` algorithm with the parameters G, u^*, ϕ, b has the following properties, [323]:

(a) If it returns a nonempty set $C \subset V$, then: (a1) $\Phi(C) \leq \phi$, hence C is a set with small conductance, and (a2) $\mathrm{vol}\, C \leq \frac{5}{6}\mathrm{vol}\,(V)$, i.e. C is not "too big" (its volume is only a fraction of the total volume of the graph). Moreover, $\mathrm{vol}\, C \geq 2^b$, i.e. C is not to small (nor too isolated) as the nodes in this set have not fewer than 2^b neighbors.

(b) If Z is a set of nodes satisfying the constraints

$$\mathrm{vol}\,(Z) \leq \frac{2}{3}\mathrm{vol}\,(V), \quad \Phi(S) \leq f_1(\phi)$$

then there exists its subset Z', such that $\mathrm{vol}\,(Z') \geq \frac{1}{2}\mathrm{vol}\,(S)$. Further, if `Nibble`$(G, v, \phi, b)$ returns a nonempty set C, where $v \in Z'$, then $\mathrm{vol}\,(Z \cap Z') \geq 2^{b-1}$.

A reader interested in implementational details is referred to the original report [323]. We should mention that, in agreement with its name, the algorithm can be used for clustering the nodes of the graph: after we obtain a nonempty set $C \subset V$, we delete it from V, together with the corresponding edges and repeat the whole procedure on the resulting diminished graph.

Although simple, the algorithm requires specification of many parameters. Moreover, the starting node u^* may lie outside the set C. The modification described in next section has no such disadvantages.

5.4.2 The `PageRank-Nibble` algorithm

The `PageRank-Nibble` algorithm uses the so-called personalized PageRank vector instead of the stochastic vector $\mathbf{p}$ from the previous section. We will denote PageRank as $\mathfrak{p}(\mathbf{s}, \alpha)$, where $\alpha \in (0, 1)$, called dumping factor, is a parameter, and $\mathbf{s}$ is the vector of the form (5.104), or, more generally,

$$\mathbf{s}(v) = \begin{cases} 1/|S| & \text{if } v \in S \\ 0 & \text{otherwise} \end{cases} \tag{5.109}$$

where S is a subset of nodes. More information on PageRank is provided in Appendix E.

Instead of making k_{max} steps during a random walk, and verifying the stoping conditions at each step, in this algorithm the vector $\mathfrak{p}(\mathbf{s}, \alpha)$, is computed first, and next a family of sets $S_j^{\mathfrak{p}(\mathbf{s},\alpha)}$, $j = 1, 2, \ldots$ is searched in order to identify the set with minimal conductance. This set represents a natural cluster, containing the nodes specified in the vector $\mathbf{s}$.

PageRank vector is nothing but a stationary distribution of a Markov chain characterized by the transition matrix

$$P = (1-\alpha)P' + \alpha \mathbf{s}\mathbf{e}^{\mathrm{T}} \tag{5.110}$$

where P' is the column-stochastic matrix defined in (5.103), $\mathbf{s}$ is the initial distribution of states, and $\alpha \in (0,1)$ is the dumping factor: $P \to P'$ if $\alpha \to 0$. The matrix P describes a random walk, in which the walker decides, with probability $(1-\alpha)$, on standard random walk, and with probability α she/he returns to a node specified in the set S (and thereafter continues the random walk). Thus, α is a probability of discouragement or irritation: the irritated walker starts a new random walk from a node randomly chosen from the set S.

In the standard definition, [274], the vector $\mathbf{s}$ is the uniform probability distribution over the set of nodes V. Otherwise, we talk about personalized PageRank vector, since the distribution $\mathbf{s}$ favors certain nodes with respect to the others. In the sequel, we will consider personalized distributions (5.109); in particular, S may contain only one node. The vector $\mathfrak{p}$ represents the stationary distribution. Hence, if P has the form (5.110), then it follows from the equation $\mathfrak{p} = P\mathfrak{p}$ that this vector satisfies the condition (see Remark E.1.1)

$$\mathfrak{p}(\mathbf{s},\alpha) = \alpha\mathbf{s} + (1-\alpha)P\mathfrak{p}(\mathbf{s},\alpha) \tag{5.111}$$

whence

$$\mathfrak{p}(\mathbf{s},\alpha) = \alpha\Big(\mathbb{I} - (1-\alpha)P\Big)^{-1}\mathbf{s} \tag{5.112}$$

Using the approximation $(\mathbb{I} - X)^{-1} = \sum_{j=0}^{\infty} X^j$, the above equation can be rewritten in the form[51]

$$\mathfrak{p}(\mathbf{s},\alpha) = \alpha\mathbf{s} + \alpha\sum_{j=0}^{\infty}(1-\alpha)^j P^j \mathbf{s} \tag{5.113}$$

Other methods of computing personalized PageRank $\mathfrak{p}(\mathbf{s},\alpha)$ are mentioned in Appendix E.2.

Having the vector $\mathfrak{p}(\mathbf{s},\alpha)$ we determine – like in the `Nibble` algorithm – the sets $S_j^{\mathfrak{p}(\mathbf{s},\alpha)}$, and we search for the set of minimal conductance. If the graph contains a well defined structure, the algorithm will find it – see [13, Sect. 6], where the full version of the algorithm, together with its properties, is described.

Example 5.4.1 *Consider the graph* `karate`*, representing social relationships among members of a karate club, [380] – see Fig. 5.24. Black square shows the starting node #7. Grey squares denote members of the set* $S_{15}^{\mathfrak{p}}$ *with minimal conductance. Left panel of Fig. 5.25 shows the values of the vector* $\mathbf{q}$ *(corresponding to the ordering function* q *from the previous section) for the original numbering of the nodes, and right panel – the values of conductance* $\Phi(S_j)$*. Note that the*

[51] Note that P^j can be computed iteratively as $P^{j-1}P$, $j = 2, 3, \ldots$. Since P is a sparse matrix, such a multiplication can be organized efficiently even in case of large graphs.

support of the vector $\mathbf{q}$ *is whole set of nodes* V*. Note also the first local minimum on the right panel; it represents the set* $\{5, 6, 7, 11, 17\}$*, constituting a well defined micro-community.* □

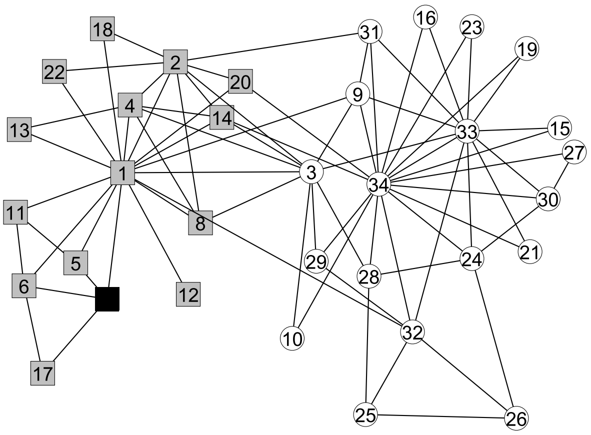


Fig. 5.24. Partition of the set of nodes of `karate` network into two groups. Black square denotes the starting node used to compute the personalized PageRank. Grey squares denote natural neighbors of the black square.

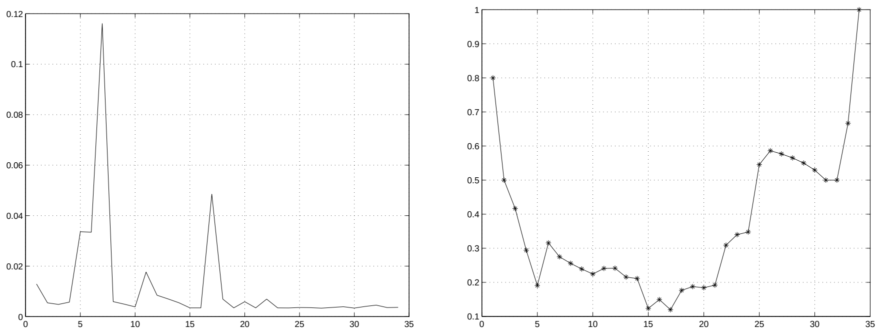


Fig. 5.25. The values of the ordering function q (left panel), and the values of conductance (right panel) computed for the `karate` network

The local methods presented in this section may be of use in many situations. Some applications of this idea are mentioned below:

- *Data provenance*[52]. Systems for tracking the origin and transformation of historical data, which create and process huge graphs. There is no need, however, to examine the entire graph; it is important to select a subset of relevant nodes only[53].
- Macropol, Can and Singh showed in [241] a modification of the `PageRank-Nibble` algorithm. It allows for the detection of natural neighbors as well as for partition of the set of nodes into (possibly overlapping) groups.
- Determination of influential pages, i.e. an indication of the webpages affecting the rank of other pages, along with an indication of the intensity of the impact, [11].
- The authors of [79] proposed a distance measure based on the order induced by the function $q(v)$. This allows for designing the advanced clustering algorithms.
- In [15] a method of identification of the densest part of a given graph is described. This idea can be used e.g. in bioinformatics to identify complex patterns in massive graphs[54].
- Adaptation of the local approach to the analysis of directed graphs is proposed in [14]. This can be used e.g. in *topic mining*[55]. Discovering thematically related document groups in massive document collections can be viewed as identifying a subgraph of a huge graph. The subgraph represents a well defined topic. Local methods allow to solve the scalability problem successfully.

5.5 Semisupervised learning

The formalism from the previous section has found interesting application is semisupervised learning. We now present two algorithms, suggested by Dengyong Zhou, [392].

The problem of semisupervised learning is as follows: let $X = \{\mathbf{x}_1, \dots, \mathbf{x}_m\}$ be a set of n-dimensional observations and $\mathbb{L} = \{-1, 0, +1\}$ be the set of labels. We assume that each observation belongs to one of the two classes, and the knowledge about observations' membership is represented by the vector $\mathbf{y}$, that is $y_i = -1$ if it is known that the observation $\mathbf{x}_i$ belongs to class C_{-1}, $y_i = 1$ if $\mathbf{x}_i$

[52] See e.g. P. Buneman, S. Khanna, W.-C. Tan: Data provenance: Some basic issues. *Proc. of 20th Conf. Foundations of Software Technology and Theoretical Computer Science*, FST TCS 2000, New Delhi, India. Springer 2000, pp. 87-93.

[53] Por. P. Macko, D. Margo, M. Seltzer: Local clustering in provenance graphs. *Proc. of the 22nd ACM Int. Conf. on Information & Knowledge Management.* ACM, 2013, pp. 835-840.

[54] B. Saha, *et al.* Dense subgraphs with restrictions and applications to gene annotation graphs. In *Research in Computational Molecular Biology*, Springer 2010, pp. 456-472

[55] S.E.G. Villarreal, R.F. Brena: Topic mining based on graph local clustering. In: I. Batyrshin, G. Sidorov, eds. *Advances in Soft Computing.* LNCS 7095, Springer 2011, pp 201-212.

belongs to class C_{+1}, and $y_i = 0$ denotes lack of knowledge about membership of $\mathbf{x}_i$ in a particular group. Suppose the objects (observations) are numbered such that we know correct membership for the first l objects, i.e. $\mathbf{y}_i \in \{-1, +1\}$ for $i = 1, \dots, l$ and $\mathbf{y}_j = 0$ for $j = l{+}1, \dots, m$. The problem consists in assigning labels to the objects numbered $j = l{+}1, \dots, m$.

When the relationships among the objects are symmetric, we can use the algorithm described in pseudocode 5.9. Elements of the similarity matrix S can be computed by using formula (5.1). We can also use the k nearest neighbors rule and set $s_{ij} = 1$ if $\mathbf{x}_j$ belongs to the neighborhood and $s_{ij} = 0$ otherwise. It is important that – like in the `NJW` algorithm from Section 5.2.5.3 – $s_{ii} = 0$ for $i = 1, \dots, m$.

The matrix $M = \mathbb{I} - \alpha\mathfrak{L}^d$ from step 3 of the algorithm is diagonally dominant, what means that: (a) it is invertible, and (b) there is a fast procedure allowing to find a vector $\mathbf{f}$ satisfying the equation $M{\cdot}\mathbf{f} = \mathbf{y}$, [322].

Algorithm 5.9 Spectral algorithm for semisupervised learning in undirected graphs, [392]

```
Require: Undirected graph $G = (X, E)$, set of $\mathbb{L} = \{-1, 0, 1\}$, the vector $\mathbf{y}$ with
    elements $y_i \in \mathbb{L}$, $i = 1, \dots, m$.
1: Create similarity matrix $S$ with entries $s_{ij}$, computed, e.g. by using equation (5.1)
    if $i \neq j$ and $s_{ii} = 0$. Let $D = \mathrm{diag}(S{\cdot}\mathbf{e})$ be the degree matrix, corresponding to $S$.
2: Determine the complement of symmetric Laplacian, $\mathfrak{L}^d = D^{-1/2} S D^{1/2}$.
3: Compute the vector $\mathbf{f} = (\mathbb{I} - \alpha\mathfrak{L}^d)^{-1}\mathbf{y}$, where $\alpha \in (0, 1)$ is a parameter.
4: return labels $\mathbf{y}_i = sgn(f_i)$.
```

Example 5.5.1 *To illustrate the effectiveness of the algorithm, three hundred two-dimensional objects were randomly generated. It was assumed that the first 120 objects belong to the class C_{-1}. Their coordinates are random values obtained from the normal distribution: $\mathbf{x}_{i,1} \sim 2 \cdot N(0, 1)$, $\mathbf{x}_{i,2} = 3 \cdot N(0, 1)$. The remaining 180 objects belong to the class C_{+1}; their coordinates are Gaussian deviates: $\mathbf{x}_{i,1} \sim 11{+}3.5{\cdot}N(0, 1)$, $\mathbf{x}_{i,2} = 10{+}3.5{\cdot}N(0, 1)$. The elements of similarity matrix were computed according to the equation (5.1), and $\sigma^2 = 2$. It was assumed that in each group 10% of objects have known label. The results of the algorithm are shown in Fig. 5.5. Circles denote objects with wrong labels. Only two such cases were noted.* □

The second algorithm, 5.10, was introduced to classify the Web pages. The pages refer to other pages, hence such a collection is modeled by a directed graph. Construction of a Markov chain with carefully chosen transition matrix allows to model the random walk in such a graph. Replacing Laplacian from the algorithm 5.9 by its directed counterpart makes it possible to design an efficient classifier. Numerical experiments described in [391] show that the entries of the vector $\mathbf{y}$ are estimated more successfully than those obtained by using a naïve symmetrization of the entire graph.

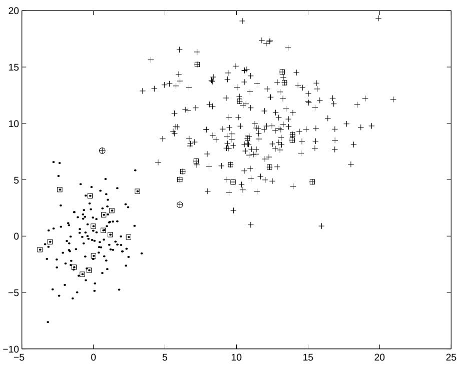


Fig. 5.26. Results of the algorithm 5.9. The members of C_{-1} group are marked by blue dots, and the members of C_{+1} – by green squares. Larger squares/circles denote the objects with labels known in advance, while red triangles – objects with wrong labels.

Algorithm 5.10 Spectral algorithm for semisupervised learning in directed graphs, [391]

```
Require: Directed graph $G = (X, E)$, set of labels $\mathbb{L} = \{-1, 0, 1\}$, vector $\mathbf{y}$ with entries $y_i \in \mathbb{L}$, $i = 1, \ldots, m$.
1: Determine the transition matrix $P$ characterizing random walk having unique stationary distribution $\pi$. Such a matrix can be computed by using the formula (D.16).
2: Compute the complement of the normalized directed Laplacian $\Theta = (\Pi^{1/2} P \Pi^{-1/2} + \Pi^{-1/2} P^{\mathrm{T}} \Pi^{1/2})/2$, where $\Pi = diag(\pi)$.
3: Determine the vector $\mathbf{f} = (\mathbb{I} - \alpha\Theta)^{-1}\mathbf{y}$, where $\alpha \in (0, 1)$ is a parameter.
4: return labels $\mathbf{y}_i = sgn(f_i)$.
```

Let us note that both algorithms can be used for hierarchical bisection of the set X if its elements are classified into more than two groups.

Another simple and efficient algorithm for semisupervised learning was proposed by Xu, Li and Schuurmans in [372]. Their algorithm solves the problems formulated as follows

$$\begin{aligned} &\max \ \mathbf{v}^{\mathrm{T}} A \mathbf{v} \\ &\text{s.t.} \quad \|\mathbf{v}\| = 1, B\mathbf{v} = c \end{aligned} \tag{5.114}$$

where A is a positively semi-defined matrix, and B is a matrix representing some constraints, e.g. known memberships of some objects. If

$$A = D^{-1/2}(D - S)D^{-1/2}$$

then, using the algorithm 5.11, it is possible to minimize the Rayleigh quotient with additional constraints.

The ideas described above were used in [393] to design a simple and (rather) universal clustering algorithm, which can be used if the objects are characterized by the feature vectors. The essence of the algorithm relies upon the ordering of objects as the inner structure of the set is discovered through the analysis of

Algorithm 5.11 Minimization of the product $\mathbf{v}^{\mathrm{T}}A\mathbf{v}$ under the constraints $\|\mathbf{v}\| = 1, B\mathbf{v} = \mathbf{c}$, [372]

```
Require: $\epsilon$ – precision
1: $t = 0$
2: $P = \mathbb{I} - B^{\mathrm{T}}(BB^{\mathrm{T}})^{-1}B$
3: $n_0 = B^{\mathrm{T}}(BB^{\mathrm{T}})^{-1}c$, $\gamma = \sqrt{1 - \|n_0\|^2}$, $v_0 = \gamma P A n_0 / \|P A n_0\| + n_0$
4: repeat
5:     $u_{t+1} = \gamma P A v_t / \|P A v_t\|$
6:     $v_{t+1} = u_{t+1} + n_0$
7:     $t = t + 1$;
8: until $\|v_t - v_{t-1}\| \leq \epsilon$
9: return vector $\mathbf{v}$.
```

its representatives. More precisely, let $X = \{\mathbf{x}_1, \dots, \mathbf{x}_m\}$ be the set of objects ordered in such a way that the first l objects have known membership in the classes, i.e.

$$y_{ij} = \begin{cases} 1 \text{ if } \mathbf{x}_i \text{ belongs to } j\text{-th class} \\ 0 \text{ otherwise} \end{cases} \tag{5.115}$$

Y is a matrix of size $m \times k$ with the entries $\mathbf{y}_{ij}$. Let S be a similarity matrix with the elements computed according to the equation (5.1) and let $\mathfrak{S} = D^{-1/2}SD^{-1/2}$ be its symmetric counterpart. Further, let

$$F^{(t+1)} = \alpha S F^{(t)} + (1-\alpha)Y \,, t = 0, 1, \dots \tag{5.116}$$

where $F^{(0)} = Y$. It was shown in [393] that $\lim_{t\to\infty} F^{(t)} = F^* = (\mathbb{I} - \alpha S)^{-1}Y$. The objects $\mathbf{x}_{l+1}, \dots, \mathbf{x}_m$ are assigned to the class, for which f_{ij} is maximal, i.e. $\mathbf{x}_i$ belongs to the class j^*, if $j^* = \arg\max_{1\leq j\leq k} f_{ij}$.

Complete analysis of this algorithm is available in [395], and its further extensions and modifications can be found in [61]. These modifications allow for:

(a) identification of the most representative elements in each cluster,
(b) identification of unusual objects in the whole collection, and
(c) identification of outstanding elements in each group.

5.6 Some improvements and other methods

In what follows, we discuss selected alternative approaches to spectral clustering.

5.6.1 Stochastic clustering

Meyer and Wessell proposed in [256] an intriguing algorithm, exploiting the properties of doubly stochastic matrices, see Lemma 2.4.1(e).

Consider an ideal situation, where the matrix $K = X^{\mathrm{T}}X$, introduced in Sect. 2.4.2.1, is block diagonal and consists of k blocks. Normalizing its rows and columns we obtain a doubly stochastic matrix P of the form

$$P = \begin{pmatrix} P_1 & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & P_2 & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & P_k \end{pmatrix}$$

All of its blocks P_j, $j = 1, \dots, k$, are doubly stochastic matrices of size $n_j \times n_j$ with *positive* entries. The eigenvalues of these blocks are positive real numbers, and exactly one, maximal eigenvalue, equals to 1. Since the spectrum of the matrix P is the union of the spectra of individual blocks, $\sigma(P) = \sigma(P_1) \cup \dots \cup \sigma(P_k)$, then it contains exactly k eigenvalues which are equal to 1.

In practice, though, the kernel matrix K is not block diagonal. Yet we can expect, that after reordering its rows and columns, it will take the form

$$K = \begin{pmatrix} K_{11} & \boldsymbol{\epsilon}_{12} & \dots & \boldsymbol{\epsilon}_{1k} \\ \boldsymbol{\epsilon}_{21} & K_{22} & \dots & \boldsymbol{\epsilon}_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{\epsilon}_{k1} & \boldsymbol{\epsilon}_{k2} & \dots & K_{kk} \end{pmatrix} \tag{5.117}$$

Here, K_{jj} is a square matrix of size $n_j \times n_j$, with rather high similarities, and $\boldsymbol{\epsilon}_{ij}$ is a matrix of size $n_i \times n_j$, whose elements do not exceed the minimal similarity value contained in the matrix K_{ii}. In other words, $\max_{k,l}(\epsilon_{ij})_{kl} \le \min_{s,t}(K_{ii})_{st}$. We call such a matrix K almost block-diagonal.

The doubly stochastic matrix P, derived from the matrix K, is also almost block-diagonal[56]. We can expect, that among the eigenvalues $1 = \lambda_1 \ge \lambda_2 \ge \dots \ge \lambda_m$, of the matrix P, the k largest eigenvalues are much greater than the remaining eigenvalues. Deuflhard and Weber[57] introduced the notion of Perron gap, defined as

$$\Delta_P = \max_{j=1,\dots,m-1} (\lambda_j - \lambda_{j+1}) \tag{5.118}$$

It is the maximum gap between two successive eigenvalues. If the maximum difference occurs between the values λ_{j^*} and λ_{j^*+1}, then the set $\{\lambda_1, \dots, \lambda_{j^*}\}$ is said to be Perron cluster in matrix P, and the number j^* is treated as an estimate of the true number of clusters in the set $\mathfrak{X}$. The bigger the value Δ_P, the more clearer structure in the dataset exists. If the same value of Δ_P was noted for several pairs of eigenvalues, then as j^* we take the lowest value of the index j.

These notions are are illustrated in Fig. 5.27. Here, matrix K consists of 3 blocks. In case (b) zeros were replaced by the the random values $0.1 \cdot \mathrm{rand}()$, and then the similarity matrix was transformed into a doubly stochastic matrix. In both cases $\Delta_P = \lambda_3 - \lambda_4$, so the Perron cluster is $\{\lambda_1, \lambda_2, \lambda_3\}$.

[56] In the Markov chain nomenclature such a matrix is referred to as *nearly uncoupled.*
[57] See papers [11] and [12], mentioned in [256].

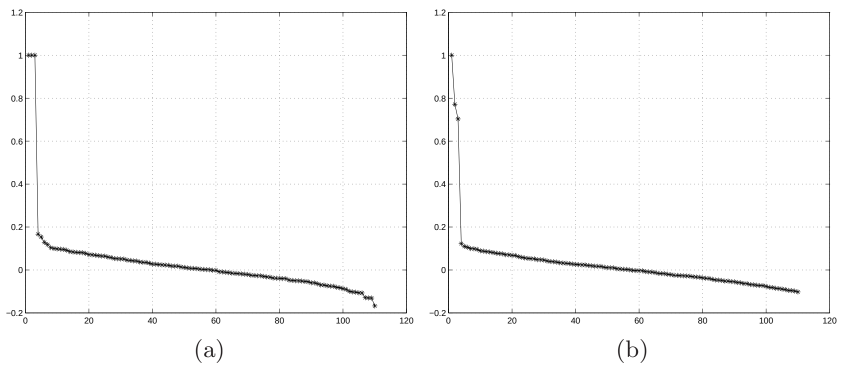


Fig. 5.27. Distribution of eigenvalues of: (a) block diagonal, and (b) almost block-diagonal doubly stochastic matrix. In case (a) $\lambda_1 = \lambda_2 = \lambda_3 = 1$, while in case (b) $\lambda_1 = 1$, $\lambda_2, \lambda_3 > 0.7$, $\lambda_4 < 0.2$.

Inspired by these facts, Meyer and Wessell proposed in [256] a simple algorithm presented below as pseudocode 5.12.

Algorithm 5.12 Stochastic clustering algorithm, [256]

```
Require: A set of data $X$; $\tau$ – a user defined parameter (integer)
Ensure: Assignment of the objects from $X$ to disjoint groups
 1: Create matrix $K$ representing similarities between objects
 2: Transform $K$ to a doubly stochastic matrix $P$
 3: Determine the eigenvalues of the matrix $P$ and evaluate the number of clusters as
    the size of the Peron cluster
 4: Initialize randomly a vector $\mathbf{x}_0$.
 5: For the successive values $t = 0, 1, \ldots$ compute the vectors $\mathbf{x}_{t+1}^{\mathsf{T}} = \mathbf{x}_t^{\mathsf{T}} P$. At each
    step, sort the values of the vector $\mathbf{x}_{t+1}$ and, guided by these values, divide the set $X$
    into $k$ disjoint sets. If the resulting partition does not change through $\tau$ subsequent
    steps, then STOP.
 6: return the partition.
```

This algorithm can be used either to cluster a given dataset, or to perform a more advanced analysis involving the aggregation of clusterings, obtained by running various algorithms. This last problem is discussed in Sect. 2.5.5.

Example 5.6.1 *Contrary to "standard" clustering algorithms, the algorithm 5.12 copes quite well with the data which are not linearly separable. Fig. 5.28(a) presents the set of 350 observations coming from three different groups. Similarity between the observations was computed by using the Gaussian kernel with the parameter* $\gamma = 2.5$. *It was assumed that the values* $k_{ij} < 0.1$ *are replaced by zeros. The doubly stochastic matrix* P*, presented in Fig. 5.28(b), was computed by using the fast algorithm, described in [207]. Since the matrix is block-diagonal, it has three eigenvalues equal to 1. The 50 largest eigenvalues are displayed (in* *descending order) in Fig. 5.28(c), and Fig. 5.28(d) shows the gaps between successive eigenvalues. Thus, the maximal gap criterion cannot be applied in this case. But the ordered values of the vector* $\mathbf{x}$*, computed in step 5 of the algorithm, suggest existence of 3 groups: for objects belonging to first group these values are in the interval* $[0.4590, 0.4871]$*, the objects from the second group have the values* $x_i = 0.4978$*, and in case of the objects from the third group, these values belong to the interval* $[0.5166, 0.5190]$*. These values are shown on Fig. 5.28(e). The final partition, obtained by analyzing the vector* $\mathbf{x}$ *is shown in Fig. 5.28(f).*

Fig. 5.29 illustrates the application of this idea to a more complicated 2-dimensional dataset. □

5.6.2 Application of Singular Value Decomposition

A serious drawback of spectral methods is constituted by the necessity to compute square similarity matrix which may in practice be very large. Shu *et al.* [314], proposed a method overcoming this problem and relying upon computing singular vectors of a rectangular matrix, characterizing the elements of the set X. Such a procedure was possible because of using a specific similarity measure.

In this approach, X is a collection of m documents, which are described by the matrix $T = (\mathbf{t}_1, \dots, \mathbf{t}_m)^{\mathsf{T}}$ of size $m \times n$. The element t_{ij} denotes the weight of j-th term in i-th document (see e.g. [29] or [246] for an explanation of the term-documents representation). Similarity between a two documents is expressed as the cosine of the angle between the vectors representing these documents. If we assume that the vectors are normalized, then the similarity of the documents equals to $s_{ij} = \mathbf{t}_i^{\mathsf{T}} \mathbf{t}_j$, see Eqn. (2.13) on p. 31. Thus, the matrix S can be expressed as the product

$$S = TT^{\mathsf{T}} \tag{5.119}$$

The degree matrix takes the form

$$D = \operatorname{diag}\big(T(T^{\mathsf{T}}\mathbf{e})\big) \tag{5.120}$$

where the j-th element of the row $\mathbf{w} = T^{\mathsf{T}}\mathbf{e}$ is computed as $w_j = \sum_{i=1}^{m} t_{ij}$. This observation allows for saving the space and time of computations, as T is a sparse matrix. Instead of determination of the square similarity matrix and summation of its rows, as done in the typical spectral approach, we sum the elements in each column of matrix T, and the resulting vector is multiplied by the matrix T.

The normalized Laplacian is computed now as follows

$$\begin{aligned} \mathfrak{L} &= \mathbb{I} - D^{-1/2} S D^{-1/2} \\ &= \mathbb{I} - D^{-1/2} T T^{\mathsf{T}} D^{-1/2} \\ &= \mathbb{I} - CC^{\mathsf{T}} \end{aligned} \tag{5.121}$$

where C stands for the matrix

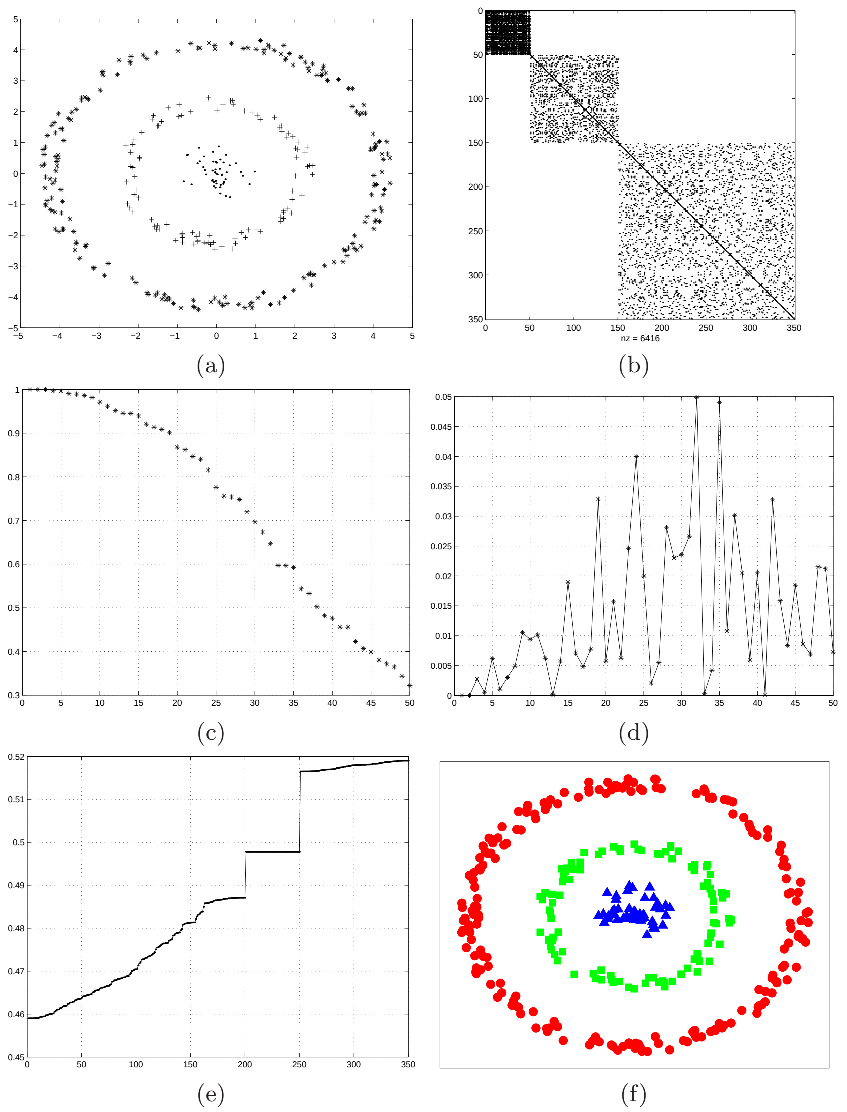


Fig. 5.28. Illustration of results produced by the stochastic clustering algorithm (a): Input data, (b): matrix P, (c) 50 maximal eigenvalues of the matrix P, (d) the values of successive gaps $\lambda_j - \lambda_{j+1}$, (e) the components of the vector $\mathbf{x}$ obtained in Set 5 of the algorithm (in ascending order), (f) partition of the set X according to the values of the vector $\mathbf{x}$

$$C = D^{-1/2}T \tag{5.122}$$

Also here we save space and time.

Consider now the SVD factorization of the matrix C

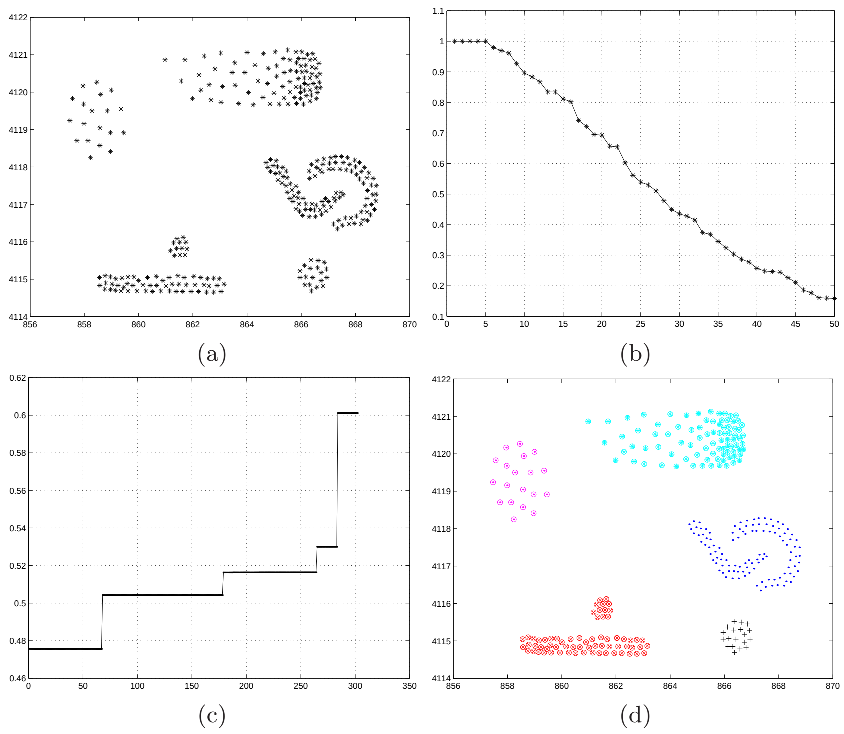


Fig. 5.29. Results obtained with the stochastic clustering algorithm. (a): Data set, (b): the values of successive gaps $\lambda_j - \lambda_{j+1}$, (c) the components of the vector $\mathbf{x}$ obtained in Set 5 of the algorithm (in ascending order), (d) partition of the set X according to the values of the vector $\mathbf{x}$

$$C = U \Sigma V^{\mathrm{T}}$$

where U and V are orthogonal matrices of the sizes, respectively, $m \times m$, and $n \times n$, and Σ is an $m \times n$ rectangular diagonal matrix. With this factorization, the Laplacian $\mathfrak{L}$ takes the form

$$\mathfrak{L} = U(\mathbb{I} - \Sigma \Sigma^{\mathrm{T}})U^{-1} \tag{5.123}$$

The matrix $\Lambda = \mathbb{I} - \Sigma\Sigma^T$ is a diagonal matrix, hence $U \Lambda U^{-1}$ defines the spectral decomposition of the Laplacian $\mathfrak{L}$. Thus, the columns of matrix U, which are singular vectors of the matrix C, are identical to the eigenvectors of the Laplacian. They can be determined using the power method discussed in [143, Ch.7.3] (see Theorem 7.3.1).

5.6.3 The PIC algorithm

This is a surprising algorithm, described by Lin and Cohen in [234]. The authors refer to the consequences of mixing time. They consider the matrix $P = D^{-1}S$

and they trace the rate with which the elements of the dominating eigenvector converge to the stationary form. In our earlier considerations we noted that this is useless! But while tracing the rate, with which elements of this vector stabilize, the authors use this information to partition the dataset[58]. More precisely, using the power method, the dominating eigenvector is computed as

$$\mathbf{x}^{(t)} = P\mathbf{x}^{(t-1)} = P^t\mathbf{x}^{(0)},\ t = 1, 2, \ldots$$

Using the property (B.11) from p. 238, this equation can be rewritten as

$$\begin{aligned}\mathbf{x}^{(t)} = P^t\mathbf{x}^{(0)} &= \sum_{i=1}^{m} \lambda_i^t \mathbf{u}_i(\mathbf{v}_i^{\mathsf{T}}\mathbf{x}^{(0)}) \\ &= \sum_{i=0}^{m} c_i \lambda_i^t \mathbf{u}_i\end{aligned} \tag{5.124}$$

where $\mathbf{u}_i$, $\mathbf{v}_i$ stand for, respectively, right and left eigenvector of the matrix P, corresponding to the eigenvalue λ_i, while $c_i = \mathbf{v}_i^{\mathsf{T}}\mathbf{x}^{(0)}$. If the eigenvalues are numbered in descending order, $1 = \lambda_1 \geq \lambda_2, \ldots, \lambda_m$ we state that

$$\frac{\mathbf{x}^{(t)}}{c_1\lambda_1} = \frac{1}{c_1}\mathbf{x}^{(t)} = \mathbf{u}_1 + \frac{c_2}{c_1}\lambda_2^t\mathbf{u}_2 + \cdots + \frac{c_m}{c_1}\lambda_m^t\mathbf{u}_m \tag{5.125}$$

Hence, the rate of convergence of the vector $\mathbf{x}^{(t)}$ to the dominating eigenvector $\mathbf{u}_1$ depends on the powered values λ_i^t. We already know that if the dataset admits a clear structure, consisting of k clusters, then $\lambda_i \approx 1$ for $i = 2, \ldots, k$, cf. [252]. Thus, after a few initial iterations, the vector $\mathbf{x}^{(t)}$ converges to the linear combination of k dominant eigenvectors, and the remaining components of the sum (5.125) vanish at the rate lower than λ_{k+1}^t. When these residual elements become sufficiently small, the vector $\mathbf{x}^{(t)}$ tends to $\mathbf{u}_1$ with an almost fixed rate. The algorithm inspired by these observations was termed *Power Iteration Clustering*, or `PIC` for short. It is illustrated by pseudocode 5.13, and few steps of this algorithm are illustrated in Fig. 5.30.

5.6.4 The `PRC` algorithm

Avrachenkov *et al.* proposed in [26] *PageRank based Clustering*, `PRC`, designed for clustering hypertext documents. The algorithm uses the natural information available, i.e. the links between the documents. So, the algorithm can be applied for clustering nodes of a directed graph.

Clustering consists here of two steps. At the first step the most influential nodes are identified. These nodes are treated as candidates for cluster centers. In the second step, the remaining nodes of the graph are assigned to appropriate clusters.

[58] We must mention here a related paper by H. Zhou, D. Woodruff: Clustering via matrix powering. In: *Proc. of the 23-rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems.* ACM, 2004. pp. 136-142.

Algorithm 5.13 The `PIC` algorithm, [234]

```
Require: Similarity matrix $S$, precision $\epsilon$
 1: Compute the stochastic matrix $P = (\mathrm{diag}(S\mathbf{e}))^{-1}S$
 2: Set $t = 0$, and initialize the vector $\mathbf{x}^{(t)}$
 3: repeat
 4:     $\mathbf{x}^{(t+1)} = \frac{P\mathbf{x}^{(t)}}{\|P\mathbf{x}^{(t)}\|_1}$
 5:     $\delta^{(t+1)} = \|\mathbf{x}^{(t+1)} - \mathbf{x}^{(t)}\|_1$
 6:     $t = t + 1$
 7: until $|\delta^{(t+1)} - \delta^{(t)}| \le \epsilon$
 8: use $k$-means algorithm on elements of $\mathbf{x}^{(t+1)}$ to get cluster assignments $C_1, \ldots, C_k$
Ensure: clusters $C_1, \ldots, C_k$
```

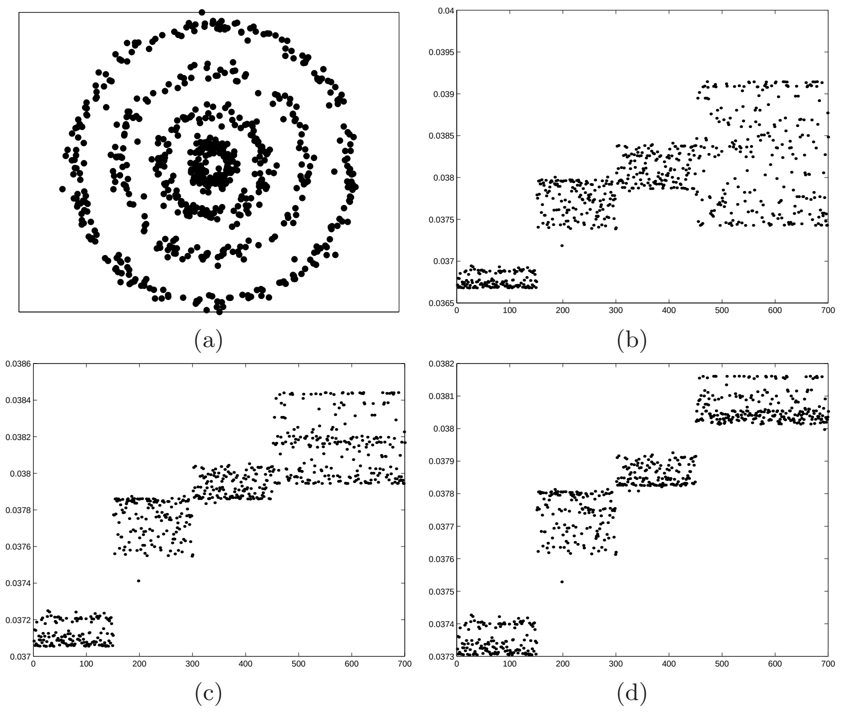


Fig. 5.30. PIC algorithm in action: (a) The dataset consisting of $k = 4$ clusters. The following figures illustrate the elements of the dominant eigenvector after: (b) – 500, (c) – 1000, (d) – 1500 iterations

The choice of groups representatives is performed in two steps, as well. The first step consistes in determining the set of the candidates, while the choice of proper candidates is done in the second step. The candidates are the nodes with high authority and simultaneously pointing at to many other pages[59]. Hence, several alternative methods can be used to create such a collection: (a) sorting the nodes in descending order with respect to the value of their PageRank, (b) sorting the nodes in descending order with respect to the product of their PageRank and Reverse PageRank, (c) sorting the nodes in descending order with respect to the HITS values [205]. Lastly, the nodes can be sorted in descending ordered with respect to their degrees. This last variant is problematic due to the ease of attaching spamming links.

Having a list of candidates, the proper nodes are selected by verifying if they belong to the same group. If so, the node with lower rank is removed. Finally, the node assignment is performed. The authors suggest to use Personalized PageRank vector to do this. Numerical experiments confirming the effectiveness of this procedure are reported in [26].

5.7 Dimensionality reduction methods

By dimensionality reduction we understand the following task: given a set X of n-dimensional observations, characterized by the (real-valued) feature vectors $\mathbf{x}_1, \ldots, \mathbf{x}_m$, create a set of n'-dimensional representatives[60] $\mathbf{y}_1, \ldots, \mathbf{y}_m$ with coordinates in space $\mathbb{R}^{n'}$, $n' << n$.

A classical method of dimensionality reduction is Principal Component Analysis (PCA). It consists in designing a *linear* mapping ϕ from $\mathbb{R}^n$ to $\mathbb{R}^{n'}$, such that the total variability of the patterns $\mathbf{y}_i = \phi(\mathbf{x}_i)$, measured in term of the sum of variances of these patterns, is maximal. This is realized by using spectral decomposition of the symmetric and positive defined covariance matrix of size $n \times n$.

The requirement of linearity is a serious limitation of this simple method. Another, also simple and popular method is Multidimensional Scaling[61]. If d_{ij}^X denotes the distance between the points $\mathbf{x}_i, \mathbf{x}_j \in X$, then we search for a set Y containing n'-dimensional points whose distances d_{ij}^Y reflect the original distances. This leads to the optimization problem

$$\min_{\mathbf{y}_1, \ldots, \mathbf{y}_m} \sum_{i \neq j} (d_{ij}^X - d_{ij}^Y)$$

In practice this problem is instantiated in many different ways, see e.g. [164, Sect. 14.8]. Also here the solution is provided by the eigenvectors of an appropriate matrix. The disadvantage of this approach results from the nature of the Euclidean distance, which can faithfully reflect only local data structure.

[59] See e.g. D. Gibson, J. Kleinberg and P. Raghavan: Inferring Web communities from link topology, *Proc. of the 9th ACM Conf. on Hypertext and Hypermedia*, p. 225-234, ACM New York 1998.

[60] i.e. the vectors saving most vital characteristic of the whole collection

[61] I. Borg, P. Groenen: *Modern Multidimensional Scaling: Theory and Applications* (2nd ed.). New York: Springer-Verlag 2005.

Both these methods lose their appeal when the data are located near, or on, a low-dimensional manifold which, is situated in the high dimensional space of observations. Such a simplified situation is illustrated in Fig. 5.31. In case (a) each observation is described by a pair of coordinates (x_{i1}, x_{i2}), but in fact the data are located on the curve forming the letter S, i.e. these data are placed on *one-dimensional* manifold embedded into 2-dimensional Euclidean space. Similarly, case (b) presents a set of observations which can be characterized by only two coordinates.

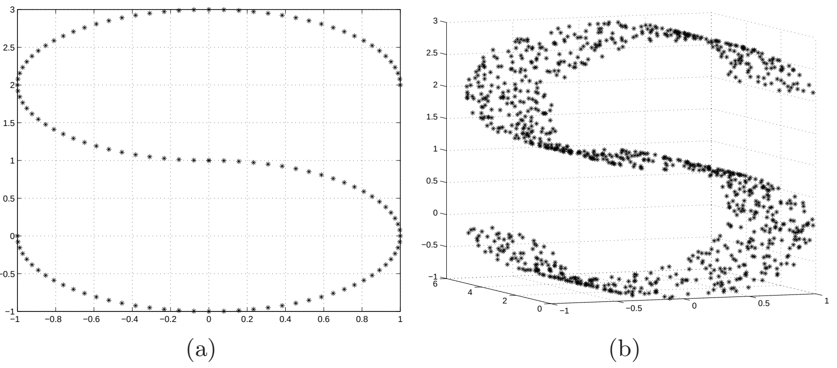


Fig. 5.31. Examples of the observations located on: (a) one-dimensional and (b) two-dimensional

Nonlinear methods of data reduction are designed to cope with such situations. Briefly speaking, they rely on "smart flatening" of the manifold, which results in the reduction of the number of coordinates necessary for precise characterization of each observation. A review of different approaches used in the so stated problem is given by Hastie, Tibshirani and Friedman in Section 14.9 of their monograph [164], and by Lee and Verleysen in [229]. A detailed discussion of these methods is beyond the scope of this book. We mention only that one of the simplest and most popular methods is *Local Linear Embedding* (LLE), developed by Roweis and Saul, and described shortly in [296]; technical details[62] can be found in [304]. The substantial difference between results returned by PCA and LLE is shown in Fig. 5.32. Panel (a) shows how particular observations are projected onto the first principal component. Note that the projection of distant original points are very close in the reduced space. In case (b), the projections reflect relationships among original data.

Another method, using spectral properties is the so-called diffusion mapping proposed by Coifman and Lafon, [83]. Suppose that S is a positive-semidefinite matrix with nonnegative entries. Let $D = \mathrm{diag}(S\mathbf{e})$ be the degree matrix computed from this S. The matrix

[62] See also `http://www.cs.nyu.edu/~roweis/lle/` for a description and source code.

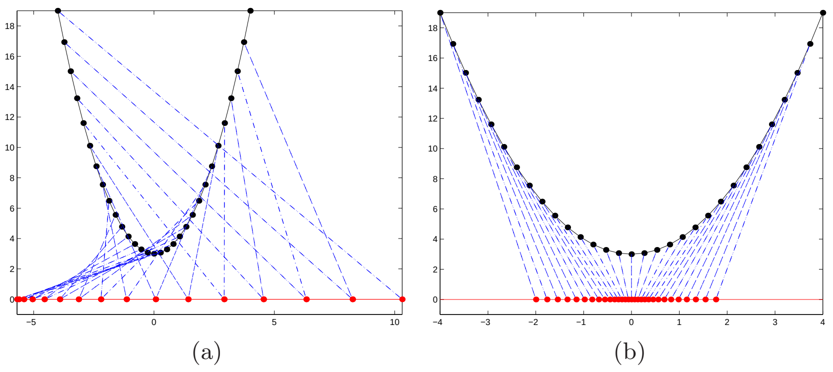


Fig. 5.32. A qualitative difference between PCA and LLE: (a) the projection obtained by PCA does not reflect the distances between observations in the original space (b) the projections produced by LLE properly represent the distances

$$P = D^{-1}S \tag{5.126}$$

is a row-stochastic matrix, and the matrix $P^{(t)} = P^t$ with elements $p_{ij}^{(t)}$ shows the probability of moving from state i to state j in t time steps. The diffusion distance between the states i, j is defined as [83]

$$D_t^2(v_i, v_l) = \sum_{l=1}^{m} \frac{\left(p_{il}^{(t)} - p_{lj}^{(t)}\right)^2}{\pi_l} \tag{5.127}$$

where π_l, $l = 1, \dots, m$ are components of the stationary distribution derived from P.

By spectral decomposition of matrix P we obtain

$$p_{ij} = \sum_{l=1}^{m} \lambda_l \psi_{il} \phi_{jl}$$

where λ_l stands for the l-th eigenvalue of the matrix P, ϕ_{jl} – denotes the j-th component of the left eigenvector of matrix P, and ψ_{il} – denotes the i-th component of l-th right eigenvector of this matrix. After some simple transformations we obtain

$$D_t^2(v_i, v_j) = \sum_{l=2}^{m} \lambda_l^{2t} \left(\psi_{il} - \psi_{jl}\right)^2 \tag{5.128}$$

Here, as usual, it was assumed that the eigenvectors are ordered according to decreasing values of the corresponding eigenvalues: $1 = \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_m$. In general, subsequent eigenvalues λ_l^{2t} vanish rather fast, what means that, with sufficient precision, we can assume that for each t there exists a number $n(t) < m$, such that

$$D_t^2(v_i, v_j) \approx \sum_{l=2}^{n(t)} \lambda_l^{2t} \big(\psi_{il} - \psi_{jl}\big)^2 \tag{5.129}$$

If $\Psi_t(v_i)$ stands for the spectral coordinates of the node (state) v_i, i.e. $\Psi_t(v_i) = (\lambda_2^{2t}\psi_{i2}, \dots, \lambda_m^{2t}\psi_{im})$, then it appears the the Euclidean distance between points with spectral coordinates is equal to their diffusion distance, that is

$$\|\Psi_t(v_i) - \Psi_t(v_j)\| = D_t(v_i, v_j) \tag{5.130}$$

Applications of this concept in data analysis, anomaly detection and knowledge discovery are discussed in [317], [258], [223].

6

Data sets

We describe below the selected data sets used throughout this book.

The sets `data3_2`, `data5_2`, and `data6_2` stem from the Web site `http://www.isical.ac.in/~sanghami/data.html`. These are 2-dimensional data sets. The first, consisting of 76 elements, breaks up into three clusters, the second one, containing 250 data points, splits into 5 clusters, and the third encompasses 300 points, forming six clusters. They are depicted in Figure 6.1.

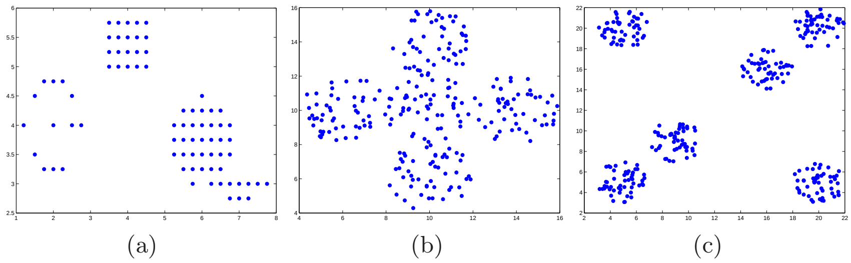


Fig. 6.1. Test data sets from the Web site `http://www.isical.ac.in/~sanghami/data.html`: (a) `data3_2`, (b) `data5_2`, (c) `data6_2`

The set `2rings`, depicted in Figure 6.2(a) is a synthetic set consisting of 200 points. 100 points lie inside a circle of radius 1 and centre at (0,0); the remaining 100 points were distributed randomly in the outer ring. By adding 100 further points forming the next ring, surrounding the two former ones, the set `3rings`, shown in Figure 6.2(b), was obtained.

The set `2spirals`, visible in Figure 6.2(c), is a set comprising 190 points. Each subset with cardinality 95 lies on the arm of one of the spirals. This set is used in many papers.

We encourage the Reader to experiment with other test data available for instance at the Web sites

- `http://www.isical.ac.in/~sanghami/data.html`,
- `http://cs.joensuu.fi/sipu/datasets/`,
- `http://www.dr-fischer.org/pub/blockamp/index.html`

Many empirical data sets can be found at the Web site `http://archive.ics.uci.edu/ml/`. We exploit in this book the set `iris`.

Social networks of varying complexity are available at many sites, including

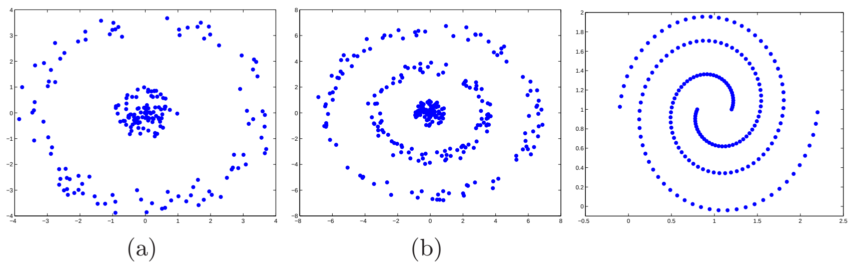


Fig. 6.2. Test data sets: (a) `2rings`, (b) `3rings`, (c) `2spirals`

- `http://deim.urv.cat/~aarenas/data/welcome.htm`,
- `https://sites.google.com/site/santofortunato/inthepress2`,
- `http://www-personal.umich.edu/~mejn/netdata/`,
- `http://www3.nd.edu/~networks/resources.htm`.

A

Justification of the FCM algorithm

We present here a justification for the formulas (3.57) and (3.58). Recall that these formulas represent a minimum of the target function.

We formulate a Lagrangian in order to determine the elements u_{ij} of the assignment matrix

$$L(J_\alpha, \lambda) = \sum_{i=1}^{m} \sum_{j=1}^{k} u_{ij}^\alpha d^2(\mathbf{x}_i, \boldsymbol{\mu}_j) - \sum_{i=1}^{m} \lambda_i \Big(\sum_{j=1}^{k} u_{ij} - 1 \Big) \tag{A.1}$$

We obtain the equation system, given below, by setting partial derivatives of the Lagrangian to zero.

$$\frac{\partial L}{\partial u_{ij}} = 0 \Leftrightarrow \alpha u_{ij}^{\alpha-1} d^2(\mathbf{x}_i, \boldsymbol{\mu}_j) = \lambda_i \ (a)$$

$$\frac{\partial L}{\partial \lambda_i} = 0 \ \Leftrightarrow \qquad \textstyle\sum_{j=1}^{k} u_{ij} = 1 \ (b)$$

The equation (a) is equivalent to the following

$$u_{ij} = \left[\frac{\lambda_i}{\alpha d^2(\mathbf{x}_i, \boldsymbol{\mu}_j)} \right]^{\frac{1}{\alpha-1}}$$

Using (b), we can write

$$\left(\frac{\lambda_i}{\alpha} \right)^{\frac{1}{\alpha-1}} = \frac{1}{\sum_{j=1}^{k} d^{2/(1-\alpha)}(\mathbf{x}_i, \boldsymbol{\mu}_j)}$$

which implies the equation (3.60).

In order to determine the components of the prototype, let us define

$$\gamma_j(\boldsymbol{\mu}_j) = \sum_{i=1}^{m} u_{ij}^\alpha \|\mathbf{x}_i - \boldsymbol{\mu}_j\|_A^2$$

Gradient of the function $\gamma_j(\boldsymbol{\mu}_j)$ with respect to the components of the vector $\boldsymbol{\mu}_j$ is of the form

$$\begin{aligned} \nabla \gamma_j(\boldsymbol{\mu}_j) &= \sum_{i=1}^{m} u_{ij}^\alpha \nabla \|\mathbf{x}_i - \boldsymbol{\mu}_j\|_A^2 \\ &= -2A \Big[\sum_{i=1}^{m} u_{ij}^\alpha (\mathbf{x}_i - \boldsymbol{\mu}_j) \Big] = 0 \end{aligned}$$

By definition of the norm, the matrix A is positive definite which guarantees the existence of the inverse matrix A^{-1}. Finally, we have

$$\boldsymbol{\mu}_j = \frac{\sum_{i=1}^{m} u_{ij}^{\alpha} \mathbf{x}_i}{\sum_{i=1}^{m} u_{ij}^{\alpha}}$$

Of course, in order to check if the solution of the optimisation task, obtained in this way, is really a (local) minimum and not a saddle point, the Hessian of the function $J_\alpha(U, M)$ needs to be investigated.

B

Matrix calculus

A detailed discussion of the topics presented here can be found in the books [255], [290], [287], [338], [298].

B.1 Vectors and their properties

We denote with the symbol $\mathbf{e}$ a vector having each component equal 1. We denote with $\mathbf{e}_j$ a vector identical with the j^{th} column of a unit matrix.

Definition B.1.1 *Let $\mathbf{x}, \mathbf{y}$ be n-dimensional vectors. They are*
(a) orthogonal, which we denote $\mathbf{x} \perp \mathbf{y}$, if $\mathbf{x}^T\mathbf{y} = 0$,
(b) orthonormal, if they are orthogonal vectors of unit length each. □

If $V = \{\mathbf{v}_1, \ldots, \mathbf{v}_k\}$ is a set of linearly independent n-dimensional vectors, where $k \leq n$, then an orthogonal set $U = \{\mathbf{u}_1, \ldots, \mathbf{u}_k\}$ of vectors, spanning the same k-dimensional subspace of the space $\mathbf{R}^n$ as V does, is obtained by application of the so-called Gram-Schmidt orthogonalisation procedure.

B.2 Matrices and their properties

Definition B.2.1 *Let $A = [a_{ij}]_{m \times m}$ be a square matrix with real-valued entries, i.e. $A \in \mathbb{R}^{m \times m}$. We will call it*
(a) non-negative (resp. positive), denoted by $A \geq 0$ (resp. $A > 0$), if all its elements are non-negative (resp[. positive)
(b) diagonal, denoted by $A = diag(d_1, \ldots, d_m)$, if $a_{ij} = 0$ wherever $i \neq j$ and $a_{ii} = d_i$, $i = 1, \ldots, m$. If all diagonal elements are identical and equal to 1, $d_i = 1$, then A is called unit matrix and is denoted with the symbol $\mathbb{I}$.
(c) symmetric, if $A = A^T$.
(d) orthogonal, if $A^T = A^{-1}$, that is, if $A^TA = AA^T = \mathbb{I}$.
(e) normal, if $AA^T = A^TA$. □

If A is an orthogonal matrix then $|\det(A)| = 1$.

Definition B.2.2 *A symmetric matrix A with dimensions $m \times m$ is called positive semidefinite, denoted by $A \succeq 0$, if for any non-zero vector $\mathbf{v} \in \mathbb{R}^m$ the following holds: $\mathbf{v}'A\mathbf{v} \geq 0$. If we can sharpen this relation ($\mathbf{v}'A\mathbf{v} > 0$ for each nonzero vector) then A is called positive definite, denoted with the symbol $A \succ 0$.*

On the other hand, if for each non-zero vector $\mathbf{v}$ $\mathbf{v}'A\mathbf{v} \leq 0$ *holds, then we call* A *negative semidefinite,. And if* $\mathbf{v}'A\mathbf{v} < 0$ *holds, then we call* A *negative definite.* □

In practice, the Sylvester theorem is used to decide on definiteness of quadratic forms. Let us denote with Δ_i, $i = 1, \ldots, m$ the leading principal subdeterminants (leading principal minors) of the matrix A, that is

$$\Delta_1(A) = a_{11},\ \Delta_2(A) = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix}, \ldots,\ \Delta_m(A) = \begin{vmatrix} a_{11} & \ldots & a_{1m} \\ \ldots & \ldots\ldots \\ a_{m1} & \ldots & a_{mm} \end{vmatrix}$$

Matrix A is positive semidefinite if $\Delta_i(A) \geq 0$, $i = 1, \ldots, m$. If, furthermore, all the leading principal minors of the matrix are positive, then it is positive definite. In the next section, see Lemma B.3.5 on page 236, we present further characterisations of positive semidefinite matrices. For completeness, let us state that if $(-1)^j \Delta_j(A) > 0$ then matrix A is negative definite, and if $(-1)^j \Delta_j(A) \geq 0$ then matrix A is negative semidefinite. It is easily seen that if A is a positive (semi)definite matrix then $B = -A$ is a negative (semi)definite matrix.

The Gram matrix is an important example of a positive semidefinite matrix.

Definition B.2.3 *If* $M = (\mathbf{m}_1, \ldots, \mathbf{m}_k)$ *is a matrix of* k *column vectors of dimension* n *then* $G = M^T M$ *is called Gram matrix.* □

The above-defined matrix G is a matrix of dimensions $k \times k$ with elements $g_{ij} = \mathbf{m}_i^{\mathrm{T}} \mathbf{m}_j = g_{ji}$. Its determinant is non-negative.

Definition B.2.4 *If a matrix* A *with dimensions* $m \times m$ *can be represented in the form* $A = BB^T$*, where* B *is a non-negative matrix of dimensions* $m \times n$*, then* A *is called a completely positive matrix*[1]*. The minimal number of columns of* B*, i.e.* n*, ensuring the above factorisation of matrix* A *is called the factorisation index or cp-rank of the matrix* A*.* □

Definition B.2.5 *The number*

$$tr(A) = \sum_{i=1}^{m} a_{ii}$$

is called matrix trace. It has the following properties:
(a) $tr(A) = tr(A^T)$,
(b) $tr(A + B) = tr(A) + tr(B)$,
(c) $tr(ABC) = tr(BCA) = tr(CAB)$ □

Definition B.2.6 *A non-negative matrix* P *is called a row-stochastic or right-stochastic matrix, if all elements of each row sum up to one. If the sum of all*

[1] See, e.g., C. Xu, Completely positive matrices, *Linear Algebra and its Applications*, 379(1), 2004, 319–327.

elements of each of its columns equals one then P is called column-stochastic or left-stochastic. If P is both column-stochastic and row-stochastic then we call it doubly stochastic matrix. □

If $A > 0$ is a symmetric matrix, then by alternating the normalising operators of rows and columns we get a doubly stochastic matrix P, [316]. Saying it differently, there exists a diagonal matrix $D = \mathrm{diag}(\mathbf{d})$ such that $P = DAD$. Components of the vector $\mathbf{d}$ are equal to $d_i = \sqrt{p_{ii}/a_{ii}}$. This method of computing the doubly stochastic matrix is called Sinkhorn-Knopp method. The paper [207] presents a quick algorithm for balancing a symmetric matrix A, that is, an algorithm transforming it into doubly stochastic matrix P, and presents a review of other related methods.

Definition B.2.7 *A stochastic matrix is called*
(a) stable, if all of its rows are identical,
(b) column-wise accessible, if its each column contains at least one positive element. □

B.3 Eigenvalues and eigenvectors

B.3.1 Basic facts

A square matrix A of dimensions $m \times m$ possesses an eigen (or characteristic) value λ and an eigen (or characteristic) vector $\mathbf{w} \neq \mathbf{0}$ if

$$A\mathbf{w} = \lambda\mathbf{w} \tag{B.1}$$

The pair $(\lambda, \mathbf{w})$, satisfying the above conditions is called eigenpair.

Equation (B.1) can be rewritten in the equivalent form

$$\det(A - \lambda\mathbb{I}_m) = 0 \tag{B.2}$$

where $\mathbb{I}_m$ is a unit matrix of dimensions $m \times m$. Knowing that formula (B.2) is a m-degree polynomial we can state that the matrix A possesses m (not necessarily distinct) eigenpairs $(\lambda_i, \mathbf{w}_i)$. If all eigenvalues are distinct then we call them non-degenerate.

The set of all distinct eigenvalues

$$\sigma(A) = \{\lambda_1, \dots, \lambda_{m'}\},\ m' \leq m \tag{B.3}$$

defines the spectrum of the matrix A, and the quantity

$$\rho(A) = \max_{1 \leq j \leq m} |\lambda_j| \tag{B.4}$$

is called spectral radius of matrix A.

If $\lambda_1, \dots, \lambda_m$ are eigenvalues of the matrix A of dimensions m, then

(a) $\sum_{i=1}^{m} \lambda_i = \mathrm{tr}\,(A)$,
(b) $\prod_{i=1}^{m} \lambda_i = \det(A)$,
(c) $\lambda_1^k, \ldots, \lambda_m^k$ are eigenvalues of the matrix A^k.

If $A = \mathtt{diag}(\mathbf{a})$ is a diagonal matrix with the diagonal identical by the vector $\mathbf{a}$, then its eigenvalues are elements of the vector $\mathbf{a}$, with the i^{th} eigenpair of the form $(a_i, \mathbf{e}_i)$, i.e. $\lambda_i = a_i$, and $\mathbf{e}_i = (0, \ldots, 0, 1, 0, \ldots, 0)^{\mathrm{T}}$ is the i^{th} column of the unit matrix.

Lemma B.3.1 *If $(\lambda_i, \mathbf{w}_i)$ is the i^{th} eigenpair of the matrix A, then eigenpairs of the matrix $c_1\mathbb{I} + c_2 A$ with c_1, c_2 being any values, for which $c_2 \neq 0$ holds, are of the form $(c_1 + c_2\lambda_i, \mathbf{w}_i)$* □

Let λ_{max} be the eigenvalue with the biggest module, i.e. $|\lambda_{max}| = \max_{i=1,\ldots,m} |\lambda_i|$ and let $\mathbf{w}_{max}$ be the corresponding eigenvector. The pair (λ_{max}, w_{max}) is called the principal eigenpair.

If we know the eigenvector $\mathbf{w}_i$ of a *symmetric* matrix A, then we can compute the corresponding eigenvalue λ_i from the equation

$$\lambda_i = \frac{\mathbf{w}_i^{\mathrm{T}} A \mathbf{w}_i}{\mathbf{w}_i^{\mathrm{T}} \mathbf{w}_i} = R(A, \mathbf{w}_i) \tag{B.5}$$

The quantity $R(A, \mathbf{x})$, defined by equation (B.5), where $\mathbf{x}$ is any non-zero vector, is called Rayleigh quotient. One can easily check that $R(A, c\mathbf{x}) = R(A, \mathbf{x})$ for any constant $c \neq 0$.

Theorem B.3.1 *Let S_k denote k-dimensional subspace of the space $\mathbb{R}^m$ and let $\mathbf{x} \perp S_k$ denote that $\mathbf{x}$ is a vector orthogonal to any vector $\mathbf{y} \in S_k$. Let $A \in \mathbb{R}^{m\times m}$ be a symmetric matrix with eigenvalues $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_m$. Then*

$$\lambda_k = \max_{S_k} \min_{0 \neq \boldsymbol{x} \perp S_k} R(A, \boldsymbol{x}) \tag{B.6}$$

□

This is the so-called Courant-Fischer minimax theorem. It implies that

$$\begin{aligned}
&(a)\ \lambda_1 = \min_{\mathbf{x}\neq 0} \frac{\mathbf{x}^{\mathrm{T}} A \mathbf{x}}{\mathbf{x}^{\mathrm{T}}\mathbf{x}}, \quad \mathbf{w}_1 = \arg\min_{\mathbf{x}\neq 0} \frac{\mathbf{x}^{\mathrm{T}} A \mathbf{x}}{\mathbf{x}^{\mathrm{T}}\mathbf{x}} \\
&(b)\ \lambda_2 = \min_{\substack{\mathbf{x}\neq \mathbf{0} \\ \mathbf{x}\perp\mathbf{w}_1}} \frac{\mathbf{x}^{\mathrm{T}} A \mathbf{x}}{\mathbf{x}^{\mathrm{T}}\mathbf{x}}, \ \mathbf{w}_2 = \arg\min_{\substack{\mathbf{x}\neq \mathbf{0} \\ \mathbf{x}\perp\mathbf{w}_1}} \frac{\mathbf{x}^{\mathrm{T}} A \mathbf{x}}{\mathbf{x}^{\mathrm{T}}\mathbf{x}} \\
&\cdots\ \cdots \qquad\qquad\qquad \cdots \\
&(c)\ \lambda_m = \max_{\mathbf{x}\neq 0} \frac{\mathbf{x}^{\mathrm{T}} A \mathbf{x}}{\mathbf{x}^{\mathrm{T}}\mathbf{x}}, \ \mathbf{w}_m = \arg\max_{\mathbf{x}\neq 0} \frac{\mathbf{x}^{\mathrm{T}} A \mathbf{x}}{\mathbf{x}^{\mathrm{T}}\mathbf{x}}
\end{aligned} \tag{B.7}$$

In this way we obtain an estimate $\lambda_1 \leq R(A, \mathbf{x}) \leq \lambda_m$.

Lemma B.3.2 *If A is a symmetric square matrix, and $\mathbf{x} \neq 0$ is a vector, then the value μ, minimising the expression $\|A\mathbf{x} - \mu\boldsymbol{x}\|$, is equal to the Rayleigh quotient.* □

Let us mention some important properties of the eigenvectors:

(i) If A is a Hermitian matrix (in particular, a symmetric one with real-valued elements) then all its eigenvalues are real numbers.
(ii) If $0 \in \sigma(A)$, then A is a singular matrix (the one with determinant equal zero)
(iii) A square matrix A with positive elements has exactly one real-valued principal eigenvalue and all the elements of the corresponding eigenvector are of the same sign (Perron-Frobenius theorem).
(iv) Eigenvectors of a normal matrix with non-degenerate eigenvalues form a complete set and are orthonormal. This means that they are versors spanning an m dimensional vector space.
One applies Gram-Schmidt orthogonalisation in case of degenerate eigenvalues. In this way one can find a set of eigenvectors that form a complete set and are orthonormal.
(v) (Gershogorin theorem, [255, Ex. 7.1.4]) Each eigenvalue λ of a square matrix A of dimension m fulfils at least one of the inequalities:

$$|\lambda - a_{ii}| \leq r_i = \sum_{\substack{j \neq i \\ 1 \leq j \leq m}} |a_{ij}|$$

i.e. λ lies inside at least one (complex) circle with a centre at the point a_{ii} and radius r_i being the sum of absolute values of non-diagonal elements of the i^{th} row. □

Definition B.3.1 *Matrices A and B are called similar, which we denote $A \approx B$, if there exists a non-singular matrix X such that*

$$A = XBX^{-1} \tag{B.8}$$

The mapping transforming the matrix B into the matrix A is called the similarity mapping. □

Lemma B.3.3 *If A and B are similar matrices then they have identical eigenvalues, and their eigenvectors satisfy the condition $\mathbf{w}_B = X^{-1}\mathbf{w}_A$.*

Proof: *Let $(\lambda, \mathbf{w}_A)$ be an eigenpair of the matrix A, i.e. $XBX^{-1}\mathbf{w}_A = \lambda\mathbf{w}_A$ (because $A \approx B$). Let us multiply both sides of this equation by X^{-1}. Then we obtain $B(X^{-1}\mathbf{w}_A) = \lambda(X^{-1}\mathbf{w}_A)$, which implies the thesis.* □

Definition B.3.2 *The square matrix A is called a diagonalisable matrix if and only if there exists a non-singular matrix X of the same dimension as A, such that $X^{-1}AX$ is a diagonal matrix. If, furthermore, X is an orthogonal matrix, i.e. $X^{-1} = X^T$, then A is called an orthogonally diagonalisable matrix.* □

Let $(\lambda_i, \mathbf{w}_i)$, $i = 1, \dots, m$ denote the set of eigenpairs of the matrix A. The matrix is diagnosable if its eigenvectors are linearly independent [2]. By substituting $X = (\mathbf{w}_1, \dots, \mathbf{w}_m)$, i.e. by placing eigenvectors of the matrix A in the columns of the matrix X we obtain $X^{-1}AX = \mathrm{diag}(\lambda_1, \dots, \lambda_m) = \Lambda$.

Lemma B.3.4 *A square matrix A is a symmetric matrix if and only if it is orthogonally diagonalisable.*

Proof: *Let us restrict our considerations to the simpler necessary condition. If A is an orthogonally diagonalisable matrix then there exists such an orthogonal matrix X and a diagonal matrix D that $A = XDX^T$. Hence $A^T = (XDX^T)^T = A$*
$\square$

Let us consider again the positive semi-definite matrices that were introduced in the preceding section.

Lemma B.3.5 *The following conditions are equivalent for a symmetric matrix A:*
(a) $A \succeq 0$.
(b) All eigenvalues of the matrix A are non-negative.
(c) $A = C^TC$, where $C \in \mathbb{R}^{m\times n}$.

Proof: *We will show that (a) implies (b), (b) implies (c), and (c) implies (a).*

(a) $\Rightarrow$ (b): As A is a symmetric matrix, hence all, its eigenvalues are real numbers. Let $(\lambda, \mathbf{w})$ be an eigenpair bound by the equation $A\mathbf{w} = \lambda\mathbf{w}$. Let us perform left multiplication of both sides of this equation with the vector $\mathbf{w}^T$. We obtain $\mathbf{w}^TA\mathbf{w} = \lambda\mathbf{w}^T\mathbf{w}$. Both the left side of this equation and $\mathbf{w}^T\mathbf{w}$ are non-negative numbers, hence $\lambda \geq 0$.

(b) $\Rightarrow$ (c): Let $\Lambda = diag(\lambda_1, \dots, \lambda_m)$ and let $W = (\mathbf{w}_1, \dots, \mathbf{w}_m)$ be a matrix with columns being eigenvectors corresponding to eigenvalues $\lambda_1, \dots, \lambda_m$. The equation (B.1) can be rewritten in matrix form as $AW = W\Lambda$. By performing right multiplication of this equation with matrix W^T we obtain $AWW^T = W\Lambda W^T$. But eigenvectors of a symmetric matrix A are orthonormal, hence $A = W\Lambda W^T$. As $\lambda_i \geq 0$, hence matrix $\mathbf{\Lambda}$ can be represented as $\Lambda^{1/2}(\Lambda^{1/2})^T$. Therefore

$$A = W\Lambda^{1/2}(\Lambda^{1/2})^TW^T = W\Lambda^{1/2}(W\Lambda^{1/2})^T = C^TC$$

where $C^T = W\mathbf{\Lambda}^{1/2}$.

(c) $\Rightarrow$ (a): Because of $A = C^TC$, we have, for any non-zero vector $\mathbf{v}$,

$$\mathbf{v}^TA\mathbf{v} = \mathbf{v}^TC^TC\mathbf{v} \Leftrightarrow \mathbf{v}^TA\mathbf{v} = \mathbf{y}^T\mathbf{y} \Leftrightarrow \mathbf{v}^TA\mathbf{v} \geq 0$$

where $\mathbf{y} = C\mathbf{v}$. $\square$

[2] Because then its rank is equal m, which is a necessary condition for the existence of the matrix inverse to the given matrix A.

Let $A_k = [a_{ij}]$, $i, j \in \{1, \ldots, k\}$ denote a submatrix of the positive semi-definite matrix A. Then, for each value $1 \leq k \leq m$ and each non-zero vector $\mathbf{x} \in \mathbb{R}^k$, the following conditions hold : (a) $\mathbf{x}^{\mathsf{T}} A_k \mathbf{x} \geq 0$ and (b) $\det(A_k) \geq 0$.

B.3.2 Left- and right-hand eigenvectors

The eigenvectors $\mathbf{w}$ of a matrix A that we have been talking about so far are called also right eigenvectors because they stand to the right of the matrix A in the defining equation (B.1). In an analogous way also the left eigenvectors can be defined.

$$\mathbf{u}^{\mathsf{T}} A = \lambda \mathbf{u}^{\mathsf{T}} \tag{B.9}$$

By transposing this equation we see that the left eigenvector of the matrix A is the right eigenvector of the matrix A^{T}. It is obvious that the distinction between left and right eigenvectors would be pointless if A were symmetric, because $\mathbf{u} = \mathbf{w}$, that is- the left and right eigenvectors are identical. Hence, let us subsequently consider predominantly the non-symmetric or even non-normal matrices. Such a distinction is not necessary for eigenvalues, because due to equation (B.2) and due to $\det(A) = \det(A^{\mathsf{T}})$ we see that in both cases one obtains the same eigenvalue.

By reasoning like in the proof of the Lemma B.3.3 we infer

Lemma B.3.6 *If A, B are similar matrices, then they have identical eigenvalues and their left eigenvectors are linked by the interelation $\mathbf{u}_B = X\mathbf{u}_A$.*

Lemmas B.3.3 and B.3.6 imply that for similar matrices A and B with A, being symmetric, the left ($\mathbf{u}_B$) and right ($\mathbf{w}_B$) eigenvectors of the matrix B can be expressed in terms of the eigenvectors of $\mathbf{v}_A$ of the matrix A as follows:

$$\mathbf{u}_B = X\mathbf{v}_A, \ \mathbf{w}_B = X^{-1}\mathbf{v}_A$$

where X is a matrix of coefficients.

Lemma B.3.7 *Let A be a matrix of dimension m possessing m distinct left and right eigenvectors*[3] *$\mathbf{u}_i$, $\mathbf{w}_i$. Then*

[3] Eigenvector normalization footnote: We deliberately postponed the issue of normalization of the eigenvectors till this point because it becomes only here clear why some choices are done. One should be aware that if a vector $\mathbf{w}$ is a right eigenvector of a matrix A, according to equation (B.1), then also any vector $c \cdot \mathbf{w}$ for any non zero scalar c is. Similarly, if a vector $\mathbf{u}$ is a left eigenvector of a matrix A, according to equation (B.9), then also any vector $c \cdot \mathbf{u}$ for any non zero scalar c is. To get rid of such an ambiguity, we assume throughout this book, if not stated otherwise, that the vectors are normalised, that is, we select that right eigenvector $\mathbf{w}$, for which $\mathbf{w}^{\mathsf{T}}\mathbf{w} = 1$ and the left eigenvector $\mathbf{u}$ having the corresponding normalised right eigenvector $\mathbf{w}$, for which $\mathbf{u}^{\mathsf{T}}\mathbf{w} = 1$. Note that the right and left eigenvectors are normalised differently.

$$A = \sum_{i=1}^{m} \lambda_i \mathbf{w}_i \mathbf{u}_i^T \tag{B.10}$$

The above spectral representation implies the equation

$$A^n = \sum_{i=1}^{m} \lambda_i^n \mathbf{w}_i \mathbf{u}_i^{\mathrm{T}} \tag{B.11}$$

In particular, if A is a symmetric matrix, we get

$$A^n = \sum_{i=1}^{m} \lambda_i^n \mathbf{v}_i \mathbf{v}_i^{\mathrm{T}} \tag{B.12}$$

where $\mathbf{v} = \mathbf{u} = \mathbf{w}$.

B.3.3 Determining eigenvalues and eigenvectors

Quick algorithms for computation of eigenvectors are indispensable for the spectral clustering methods. Many monographs have been devoted to this topic, e.g. [31], [298], [338]. Projection methods seem to play a key role here. Chapter 9 of the monograph [143] is devoted to them.

Here below we present the simplest method allowing to find quickly the principal eigenvector and then discuss its application for determining eigenpairs of the Laplacians.

B.3.3.1 The power method

As stated previously, eigenvectors $\mathbf{x}_1, \dots, \mathbf{x}_m$ of a diagonalisable matrix A are linearly independent. Therefore, they constitute a vector base in the space $\mathbb{R}^m$. Any vector $\mathbf{x}^{(0)} \in \mathbb{R}^m$ can be represented in the form

$$\mathbf{x}^{(0)} = c_1 \mathbf{x}_1 + \cdots + c_m \mathbf{x}_m$$

where $c_1, \dots, c_m$ are scalars. By multiplying the above equation by A^k, $k = 1, 2, \dots$, we get

$$\begin{aligned} A^k \mathbf{x}^{(0)} &= c_1(A^k \mathbf{x}_1) + \cdots + c_m(A^k \mathbf{x}_m) \\ &= c_1(\lambda_1^k \mathbf{x}_1) + \cdots + c_m(\lambda_m^k \mathbf{x}_m) = c_1 \lambda_1^k \Big[\mathbf{x}_1 + \sum_{j=2}^{k} c_j \Big(\frac{\lambda_j}{\lambda_1}\Big)^k \mathbf{x}_j \Big] \end{aligned} \tag{B.13}$$

We exploit here the fact that $A\mathbf{x}_j = \lambda_j \mathbf{x}_j$, where λ_j denotes the eigenvalue corresponding to the eigenvector $\mathbf{x}_j$, and that whenever λ_j is an eigenvalue of the matrix A, then λ_j^k is an eigenvalue of the matrix A^k. If eigenvalues are sorted according to their decreasing module, then $|\lambda_j/\lambda_1| \leq 1$. So, if only $|\lambda_j/\lambda_1| < 1$ then

$$\lim_{k\to\infty} \frac{\mathbf{x}^{(k)}}{\lambda_1^k} = \lim_{k\to\infty} \frac{A^k\mathbf{x}^{(0)}}{\lambda_1^k} = c_1\mathbf{x}_1$$

that is, the series $\{\mathbf{x}^{(k)}/\lambda_1^k\}$ converges with speed dependent on the quotient $|\lambda_2/\lambda_1|$, to the vector $c_1\mathbf{x}_1$.

In practice, the successive approximations of the eigenvector are not computed from the equation $\mathbf{x}^{(k)} = A^k\mathbf{x}^{(0)}$, but rather iteratively, $\mathbf{x}^{(k)} = A\mathbf{x}^{(k-1)}$, $k = 1, \ldots$. In this way we do not need to compute successive powers of the matrix A. Instead, we multiply each time the matrix A with the vector $\mathbf{x}^{(k-1)}$, obtained in the preceding step. As $\|\mathbf{x}^{(k)}\| \to 0$ when $|\lambda_1| < 1$ (or $\|\mathbf{x}^{(k)}\| \to \infty$ when $|\lambda_1| > 1$), hence, to avoid overflow and underflow, the vector $\mathbf{x}^{(k)}$ is normalised. If we denote by $\mathbf{y}^{(k)}$ the product $A\mathbf{x}^{(k-1)}$ then $\mathbf{x}^{(k)} = \mathbf{y}^{(k)}/m(\mathbf{y})$, where $m(\mathbf{y})$ is the first element of the vector $\mathbf{y}$ with the largest module. If, for example, $\mathbf{y}^{(k)} = (1, -5, 2, 5)$, then $m(\mathbf{y}^{(k)}) = -5$. Thereby not only the division error is minimized in line 6 of pseudocode B.1, but also the total computational burden is minimized, see [255, page 534], [290]. The value $m(\mathbf{y}^{(k)}) \to \lambda_1$ when $k \to \infty$, so we get also a method to determine an approximation of the principal eigenvalue. We choose usually, as the stopping criterion, the criterion of minimal correction, i.e. $\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\| < \epsilon$, or $|m(\mathbf{y}^{(k+1)}) - m(\mathbf{y}^{(k)})| \leq \epsilon$. One initialises the vector $\mathbf{x}^{(0)}$ randomly. Such an initialisation is particularly recommended if the main diagonal contains elements that are significantly larger than the other elements of the matrix A.

The method of computation of the principal eigenpair is presented as the pseudocode B.1.

Algorithm B.1 Power method returning the principal eigenpair of the matrix A

```
1: $k = 0$.
2: Initialise the vector $\mathbf{x}^{(k)}$. {$\mathbf{x}^{(k)}$ cannot be orthogonal to the principal eigenvector.
   In practice, it is sufficient that all its components have the same sign. }
3: while not done do
4:     $\mathbf{y}^{(k+1)} = Ax^{(k)}$
5:     $\beta^{(k+1)} = m(\mathbf{y}^{(k+1)})$
6:     $\mathbf{x}^{(k+1)} = \mathbf{y}^{(k+1)}/\beta^{(k+1)}$
7:     if $\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\| \leq \epsilon$ then
8:         done = true;
9:     end if
10:    $k = k + 1$
11: end while
12: return eigenvalue $\lambda \approx \beta^{(k+1)}$ and the corresponding eigenvector $\mathbf{x}^{(k)}$
```

Remark B.3.1 *The weakness of the power method is that if the principal eigenvalue is a complex number, then the approximation series $\mathbf{x}^{(k)}$ is not convergent!*
□

The power method returns the *principal* eigenpair, which means that the obtained approximation $\widetilde{\lambda}$ can be a negative number. If our goal is to find the eigenvector corresponding to the maximal *positive* eigenvalue, then we proceed as follows: We construct a matrix $A' = A - \widetilde{\lambda}\mathbb{I}$. Its eigenvalues are $\lambda'_i = \lambda_i - \widetilde{\lambda} \geq 0$. One can verify that if λ_i is the i^{th} eigenvalue of the matrix A, and $\mathbf{x}_i$ is the corresponding eigenvector then

$$(A - \widetilde{\lambda}\mathbb{I})\mathbf{x}_i = \lambda_i \mathbf{x}_i - \widetilde{\lambda}\mathbf{x}_i$$

Hence, the matrices A and A' have identical eigenvectors, while the principal eigenvalue of the matrix A' corresponds to the maximal positive eigenvalue of the matrix A increased by o $-\widetilde{\lambda}$. So, by repeating the power method, this time for the matrix A', we find the eigenvector corresponding to the maximal positive eigenvalue of the matrix A. This approach is referred to as the "deflation method".

B.3.3.2 Determining the eigenpairs of the Laplacian

In case of spectral cluster analysis, we are interested in finding $p < m$ eigenvectors, corresponding to the lowest non-trivial eigenvalues of the Laplacian $L = D - A$, where $A \in \mathbb{R}^{m \times m}$ is a symmetric matrix[4], and D is a diagonal matrix with elements $d_{ii} = \sum_{j=1}^{m} a_{ij}$. An interesting solution to this problem was proposed by Koren, Carmel and Harel in [212]. The method is outlined in the pseudocode B.2. Originally, the method was developed for drawing graphs, which means that only the first and the second (positive) non-trivial Laplacian eigenvalues are sought.

Let us note, first, that if $(\lambda_i, \mathbf{v}_i)$ are eigenpairs of the matrix A, then $(g - \lambda_i, \mathbf{v}_i)$ are eigenpairs of the matrix $\widetilde{A} = g\mathbb{I} - A$. In particular, if A is a Laplacian, and $g \geq \max_{1 \leq i \leq m} |\lambda_i|$, then by applying the deflation method one can determine the eigenpairs corresponding to the lowest eigenvalues of the Laplacian. Gershogorin theorem is used to estimate the value of g – see property (iv) from page 235, according to which the eigenvalues of a matrix A belong to the set sum of discs $\mathcal{K}_i$ in the complex plane.

$$\mathcal{K}_i = \{z \in \mathbb{C} : |z - a_{ii}| \leq \sum_{j \neq i} |a_{ij}|\}, \qquad i = 1, \dots m$$

Eigenvalues of any Laplacian $L = [l_{ij}]$ are non-negative. Furthermore $l_{ii} = \sum_{j \neq i} |l_{ij}|$. Therefore, to estimate the largest eigenvalue, it is sufficient to compute the value

$$g = 2 \cdot \max_{1 \leq j \leq m} l_{ii} \tag{B.14}$$

$L = D - A$, hence $\text{diag}(L) = \text{diag}(D)$, which allows to accelerate the computations significantly. It is the step 2. of the algorithm B.2. Let us sort the

[4] A is interpreted as a neighbourhood matrix (similarity or adjacency matrix), see Section 5.1.

eigenvalues of a Laplacian increasingly, and let λ_i be the i^{th} eigenvalue in this order. Then, values $\widehat{\lambda}_i = g - \lambda_i$ (decreasing with growing i) are the eigenvalues of the matrix $\widehat{L} = g\mathbb{I} - L$. This transformation is performed in step 3. of the algorithm B.2.

The eigenvector, corresponding to the value $\widehat{\lambda}_1 = g$, is a normalised unit vector, that is - its components are equal $1/\sqrt{m}$. Therefore, one determines the eigenvectors corresponding to eigenvalues $\widehat{\lambda}_2 \geq \ldots, \geq \widehat{\lambda}_p$ – steps 4 – 19. The initial, random approximation of the i^{th} eigenvector (step 5) is subject to Gram-Schmidt orthogonalisation, so that the resultant vector will be orthogonal to the already determined approximations of eigenvectors (step 10). Next one step of the power method is executed.

Let us look at the stopping condition of the loop **while** – **do**. Formally, one can iterate the loop till the product $\widehat{\mathbf{v}}\mathbf{v}$ exceeds the threshold $1 - \epsilon$. In practice, one observes stabilisation after a number of iterations, $\widehat{\mathbf{v}}^{\mathsf{T}}\mathbf{v} = const$. Therefore, the formal condition was replaced by a more natural condition $prev - next = 0$, where $next$ (resp. $prev$) is the current (resp. previous) value of the product $\widehat{\mathbf{v}}^{\mathsf{T}}\mathbf{v}$.

The only serious computational burden of the algorithm is determination of the product $\widehat{L}\mathbf{v} = \widehat{D}\mathbf{v} - Av$. Notice, however, that $\widehat{D}\mathbf{v}$ is a vector with components of the form $(g - \mathsf{d}_i)v_i$, and A is a sparse matrix.

Algorithm B.2 Applying power method to determine the first $p \leq m$ eigenvectors of the Laplacian

```
Require: $A$ – neighbourhood matrix, $V = [v_{ij}]_{m \times p}$ – a matrix of randomly generated
    and normalised columns; elements of the first column are of the form $v_{i1} = 1/\sqrt{m}$,
    $i = 1, \ldots, m$.
1: $L = D - A$, where $D$ is the degree matrix
2: $g = \max_{1 \leq i \leq m} \left( l_{ii} + \sum_{j \neq i} |l_{ij}| \right)$
3: $\widehat{L} = g\mathbb{I} - L$ {inversion of the order of the eigenvalues }
4: for $i = 2$ to $p$ do
5:    $\widehat{\mathbf{v}}_i = V(:, i)$ {$i^{th}$ column of the matrix $V$}
6:    $prev = 1$, $next = 0$, done = false
7:    while (not done) do
8:       $\mathbf{v} \leftarrow \widehat{\mathbf{v}}_i$
9:       for $j = 1$ to $i - 1$ do
10:         $\mathbf{v} = \mathbf{v} - (\mathbf{v}^{\mathsf{T}}\mathbf{v}_j)\mathbf{v}_j$ { Gram-Schmidt orthogonalisation}
11:      end for
12:      $\widehat{\mathbf{v}} = \widehat{L}\mathbf{v}$
13:      $\widehat{\mathbf{v}} = \widehat{\mathbf{v}}/\|\widehat{\mathbf{v}}\|$
14:      $next = \widehat{\mathbf{v}}^{\mathsf{T}}\mathbf{v}$
15:      done = $|prev - next| < \epsilon$
16:      $prev = next$
17:   end while
18:   $V(:, i) = \widehat{\mathbf{v}}$ {saving the vector $\widehat{\mathbf{v}}$ in the $i^{th}$ column of the matrix $V$}
19: end for
```

B.4 Norms of vectors and matrices

A tool allowing to measure the "size" of a vector $\mathbf{x} \in \mathbb{R}^m$ is its norm $\|\mathbf{x}\|_p$, defined as follows:

$$\|\mathbf{x}\|_p = \Big(\sum_{i=1}^{m} |x_i|^p\Big)^{1/p}, \ p = 1, 2, \ldots \tag{B.15}$$

If we set $p = 1$, then we obtain the so-called Manhattan norm (called also taxicab metric, rectilinear distance , city block distance, Manhattan distance, or Manhattan length) , $\|\mathbf{x}\|_1 = \sum_i |x_i|$, while $\|\mathbf{x}\|_2$ is called Euclidean norm, or simply – vector length. Finally, $p = \infty$ corresponds to the maximum norm, also known as supremum norm, sup norm, the Chebyshev norm, the infinity norm or the "uniform norm":

$$\|\mathbf{x}\|_\infty = \max_{1 \le i \le m} \mathbf{x}_i \tag{B.16}$$

The case of $p \in (0, 1)$ was investigated exhaustively by Aggarwal, Hinneburg and Keim in the paper [4], where they suggest its high usefulness.

Given a rectangular matrix $A \in \mathbb{R}^{n \times m}$, one defines either the Frobenius norm

$$\|A\|_F = \sqrt{\operatorname{tr}(A^{\mathsf{T}} A)} = \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{m} a_{ij}^2} \tag{B.17}$$

or one uses a matrix norm induced by a vector norm

$$\|A\|_p = \sup_{\mathbf{x} \ne 0} \frac{\|Ax\|_p}{\|\mathbf{x}\|_p} = \sup_{\|\mathbf{x}\|_p = 1} \|Ax\|_p \tag{B.18}$$

In particular:

(i) $\|A\|_1 = \max_{1 \le j \le m} \sum_{i=1}^{n} |a_{ij}|$ is the maximum value of the sum of modules of column elements,
(ii) $\|A\|_2 = \sqrt{\lambda_{max}}$, where λ_{max} is the maximal eigenvalue of the matrix $A^{\mathsf{T}} A$; this norm is called spectral norm,
(iii) $\|A\|_\infty = \max_{1 \le i \le n} \sum_{j=1}^{m} |a_{ij}|$ is the maximum value of the sum of modules of row elements.

If A is a diagonal matrix, $A = \operatorname{diag}(a_1, \ldots, a_m)$, then

$$\|A\|_p = \max_{1 \le j \le m} |a_j|, \ p = 1, 2, \ldots$$

More information on this subject can be found for example, in chapter 5 of the monograph [255].

C

Graph theory

C.1 Basic definitions

A graph G is a pair of sets (V, E), where V is called the set of vertices (or nodes) and E is a set of edges (or links, because they are deemed to link the nodes). Depending on the interpretation of the set E we speak about directed or undirected graphs. In a directed graph $E \subseteq V^2$. In an undirected graph E is a subset of the set of all subsets of V with cardinality 2 (or 2 and 1 if we allow for loops in the graph). Thus, the ordering of nodes in an edge is important for a directed graph and unimportant for an undirected one. A graph may be labelled either with respect to nodes, or to edges, or both, if we define functions $f_V : V \to S$, $f_E : E \to S$, or both, where S is a set. In particular, if the function f_E is defined in such a way that S is a (sub)set of the natural numbers, then we speak about a multigraph (a graph with multiple edges between two nodes) which reduces to a plain graph if $S = \{1\}$. If the function f_E is defined in such a way that S is a (sub)set of the positive real numbers (zero is sometimes permitted to denote a non-existent edge), then we speak about a weighted graph (a graph with differing strengths of connections between nodes). Again, it reduces to a plain graph if $S = \{1\}$.

The cardinality of the set V is called the order of the graph , and the cardinality of the set E is called the size of the graph. One defines also the density of a graph as the number g equal to

$$g = \frac{2\mathfrak{m}}{|V|(|V| - 1)} \tag{C.1}$$

for undirected graphs and

$$g = \frac{\mathfrak{m}}{|V|(|V| - 1)} \tag{C.2}$$

for directed graphs.

Subsequently, we discuss, first, the concepts relevant to undirected graphs, and later on we present their counterparts for directed graphs.

C.1.1 Undirected graphs

(1) We say that the edge $e = (u, v) \in E$ connects the nodes u and v, and the nodes u and v are called end points of the edge e.[1] Nodes connected by an edge are called neighbours (they are said to be adjacent). If $e = (u, u) \in E$, meaning that u is its own neighbour, then the edge e is called a self-loop. A graph without self-loops and without multiple edges is called a simple graph. Subsequently, we consider only simple graphs.

(2) A graph $G^c = (V, E^c)$ is called a complement (or inverse) graph of the graph $G = (V, E)$, if no edge $e \in E$ belongs to E^c and each edge $e' \notin E$ belongs to E^c.
A subset of nodes $V' \subset V$, in which no two nodes are neighbours of one another, is called an independent set (or stable set) in the graph G.
If $D \subset V$ is such a set of nodes that each node $v \in V \backslash D$ is a neighbour of a node from the set D, then D is called a dominating set in the graph G.

(3) The set of neighbours of the node u, denoted with the symbol $N(u)$, is defined as

$$N(u) = \{v \in V \colon (u, v) \in E\} \tag{C.3}$$

and its cardinality $|N(u)|$ is called the degree of the node u and we denote it with the symbol d_u. If $|N(u)| = 1$, then the node u is called a leaf or a pendant vertex. Edges connecting leaves with their neighbours are called pendant edges.

(4) A graph, in which each node has the same number of neighbours, $\mathsf{d}_i = d$, $i = 1, \ldots, |V|$ is called a d-regular graph. In particular:

- If $d = 2$, then the graph is called a graph cycle and is denoted with the symbol C_m, where m is the graph order (i.e. the number of its nodes). By removing a single edge from this graph we obtain a linear graph, denoted with the symbol P_m (it is not a regular graph).
- If $d = |V| - 1$, then the graph G is called a complete graph or a clique and is denoted with the symbol K_m, where $m = |V|$.

We distinguish one important class of d-regular graphs, called "expanders". Expanders are relatively sparse graphs, in which each "small" subset of vertices possesses a "large" neighbourhood. We say that a d-regular graph c-expands, or is a (d, c)-expander, if for any set of vertices S with cardinality not bigger than $|V|/2$ the condition $|N(S) \backslash S| \geq c|S|$ holds, where $N(S) = \bigcup_{v \in S} N(v)$.

(5) We speak about expansion of (general) undirected graphs, which is understood as the number

[1] Strictly speaking, we shall write that $e = \{u, v\}$ in the case of undirected graphs, but we will use the popular convention (u, v) understanding that $(u, v) = (v, u)$ in the context of undirected graphs.

$$\alpha(G) = \min_{\emptyset \neq S \subset V} \frac{|\partial S|}{\min(|S|, |V| - |S|)} \tag{C.4}$$

where the symbol ∂S means the set of edges, for which one end belongs to the set S and the other to its complement $\overline{S}$.

A measure indicating the degree, to which the neighbours of a node u resemble a clique, is the so-called local clustering coefficient, defined as the ratio of the number of edges linking pairs of nodes being neighbours of the node u to the number of all possible edges connecting them [360]

$$c_u = \frac{2|\{(v_i, v_j) \in E \colon v_i, v_j \in N(u)\}|}{\mathsf{d}_u(\mathsf{d}_u - 1)} \tag{C.5}$$

(6) Let $V = \{v_1, \dots, v_m\}$ be a set of nodes. If each edge $(v_i, v_j) \in E$ is assigned a weight s_{ij}, then $G = (V, E)$ is called a weighted graph. In a special case one can assume that

$$s_{ij} = \begin{cases} 1 \text{ when } (v_i, v_j) \in E \\ 0 \text{ otherwise} \end{cases} \tag{C.6}$$

The concept of node degree in an unweighted graph has its counterpart within the realm of weighted graphs, called the strength (or typicality) of a node, which is defined as the sum of weights of edges linking node u to its neighbours. If s_{ij} is interpreted as node similarity then d_i expresses the "typicality" of the node i, that is - its total similarity to all the other nodes.

(7) We call a path in a graph G a sequence of nodes $d(v_0, v_l) = v_0, v_1, \dots, v_l$ such that $(v_{i-1}, v_i) \in E$ for each $i = 1, \dots, l$. A path $v_0, v_1, \dots, v_l$ is said to connect nodes v_0 and v_l; these nodes are called end nodes of the path. We say also that the node v_l is reachable from the node v_0, and the number l is called the length of this path. If $v_l = v_0$, then the path $v_0, v_1, \dots, v_l$ is called a closed path or a cycle.
If d, d' are two paths sharing only the end points then these paths are called independent.

(8) The concept of path enables the introduction of concepts of graph distance and connectivity. A graph is called connected if for any two nodes there exists at least one path connecting them. The path of minimal length connecting two nodes u and v is called a geodesic, and the length $d(u, v)$ of this geodesic is called the distance between the nodes at its ends. If no path exists connecting u and v, then we assume $d(u, v) = \infty$. The maximum distance between any pair of nodes of a connected graph G is called graph diameter; the diameter is denoted with $diam(G)$, i.e. $diam(G) = \max_{u,v \in V} d(u, v)$.
If S denotes the neighbourhood matrix of the graph G (i.e. $s_{ij} \in \{0, 1\}$), and if N is the lowest number such that S^N is a positive matrix, then $diam(G) = N.$,

C.1.2 Directed graphs

The definitions from the previous section need a refinement if they should be applied to directed graphs. We present below such refinements of selected definitions. As in previous section, we consider only simple graphs.

(1) In a directed graph, we say that the edge $e = (u, v)$ is directed from the node u to the node v, which may be denoted as $u \to v$. The node u is called the head of the edge and the node v is called the tail of the edge. Instead of saying "edge" one frequently uses the term "arc", "directed edge" or "arrow".

(2) A path in a directed graph G is such a sequence of nodes $\{v_0, v_1, \ldots, v_l\}$, that $(v_{i-1}, v_i) \in E$ for $i = 1, \ldots, l$. The graph G is called strongly connected if for each pair of nodes s, t there exists a path of length $l \geq 1$ such that $s = v_0$, $t = v_l$.
If $v_0 = v_l$ then the path is called a (directed) cycle.
A graph G is called aperiodic, if there exists no such natural number $k > 1$, which is a divisor of the length of all of its cycles[2]. Otherwise, we can also say that G is an aperiodic graph if the largest common divisor of the lengths of all its cycles is equal 1.

(3) Let s_{ij} denote the weight of the arc $v_i \to v_j$. The simplest case is when $s_{ij} = 1$ whenever $v_i \to v_j \in E$. For each node we define its in-degree d^- and its out-degree d^+

$$\mathsf{d}_j^- = \sum_{\substack{v_i \in X \\ v_i \to v_j}} s_{ij} \tag{C.7}$$

$$\mathsf{d}_j^+ = \sum_{\substack{v_l \in X \\ v_j \to v_l}} s_{jl} \tag{C.8}$$

C.2 Graph matrices

A convenient characterisation of a graph is ensured by its neighbourhood matrix $A = [a_{ij}]$ with entries

$$a_{ij} = \begin{cases} 1 \text{ if } (v_i, v_j) \in E \\ 0 \text{ otherwise} \end{cases} \tag{C.9}$$

It is a symmetric matrix in the case of undirected graphs. Then, the degree of the i^{th} node is the sum of the elements of the i^{th} column of the matrix A. In case of weighted graphs it can be replaced by the matrix S, mentioned in definition (6) from section C.1.1.

Knowing the neighbourhood matrix we can determine so-called left (or column) stochastic matrix P by dividing elements of each column by its degree,

[2] By this definition an acyclic graph is a periodic graph.

$p_{ij} = a_{ij}/\mathsf{d}_j$. It is the transition matrix of a certain Markov chain, describing random walk in the graph G. The so-called transfer-matrix (called also the right or row stochastic matrix) is of the form $T = P^{\mathrm{T}}$.

A graph spectrum G is understood as the set of distinct eigenvalues of its neighbourhood matrix. Spectral properties of a graph play an important role in the graph theory, see e.g. the monograph [62] or [75]. In particular, in order to determine the time needed to reach a stationary distribution in the random walk process, it is sufficient to compute the principal eigenvector of the matrix T.

Many applications benefit from the combinatorial Laplacian (called simply Laplacian) L of the graph G, defined as

$$L = D - A \tag{C.10}$$

where D is a diagonal matrix, having the elements equal to the degrees of nodes of this graph. We present below the selected properties of Laplacians of undirected and directed graphs. A reader interested in further applications of the Laplacian is advised to consult the papers [259, 260].

C.2.1 Laplacian of a graph

Equation C.10 defines the combinatorial (or unnormalised) Laplacian of an undirected graph. We discuss its properties below along with those of the so-called normalised Laplacian. We will also introduce respective definitions applying to directed graphs.

C.2.1.1 Laplacian of an undirected graph

Let us summarise its basic properties

Lemma C.2.1 *Combinatorial Laplacian L possesses the following properties:*
(a) L is a symmetric and positive semi-definite matrix
(b) Sum of elements of each row and of each column of a Laplacian is equal zero.
(c) L has m non-negative real-valued eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_{m-1}$.
(d) The lowest eigenvalue $\lambda_1 = 0$ of the Laplacian corresponds to a constant eigenvector $\mathbf{e}$. □

The positive semi-definiteness of the Laplacian results from the identity[3] demonstrated in section 5.2.1.1:

$$\mathbf{x}^{\mathrm{T}} L \mathbf{x} = \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} s_{ij} (x_i - x_j)^2 \tag{C.11}$$

where $\mathbf{x} \neq \mathbf{0}$ is any vector. As L is a symmetric matrix, then its eigenvalues are real numbers. Because of positive semi-definiteness the eigenvalues are non-negative,

[3] see equation (5.16) on page 152.

$\lambda_1 = 0$ is the lowest value of a Laplacian, which is an immediate consequence of the fact (b) in the above Lemma. Therefore, L is a singular matrix, so an inverse matrix of a Laplacian does not exist. However, a pseudo-inverse can be computed. We discuss this topic in section C.2.1.2.

The lowest positive (the second) eigenvalue of the Laplacian, that is - the Fiedler value carries the information on the connectivity of the graph G. Namely

Lemma C.2.2 *[123] Let $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_m$ be eigenvalues of the Laplacian L of the graph G. This graph is connected if and only if $\lambda_2 > 0$. In case of a not connected graph, the number of its connected sub-graphs is equal to the multiplicity of the eigenvalue λ_1.* □

The next Lemma, mentioned as Conclusion 3.2 in the original paper by Fiedler [122], points at an important property of the eigenvalue λ_2.

Lemma C.2.3 *Let $G_1 = (X, E_1)$ be a sub-graph of the graph $G = (X, E)$, i.e. both graphs have a common set of vertices, but the graph G_1 contains only a subset of edges of the graph G, $E_1 \subset E$. Let λ_2^G and $\lambda_2^{G_1}$ be the second lowest eigenvalues of the Laplacian of the respective graphs. Then*

$$\lambda_2^{G_1} \leq \lambda_2^G \tag{C.12}$$

□

This implies that removal of edges from the graph G, hence the reduction of its connectivity, is accompanied by the decrease of eigenvalue λ_2 of its Laplacian. Therefore, the Fiedler value λ_2 is called algebraic connectivity of the graph G. The paper [92] may be consulted for a broader overview of properties of the Fiedler value.

The normalised Fiedler vector $\mathbf{v}_2$ and the Fiedler value λ_2 are related as follows

$$\lambda_1 = v_1^{\mathrm{T}} L v_1 = \sum_{(i,j)\in E} (v_{1,i} - v_{1,j})^2 \tag{C.13}$$

Definition C.2.1 *Conductance of the cut $\{C, \overline{C}\}$ of the graph G is defined as*

$$\phi_G(S) = \frac{vol\,\partial C}{\min(vol C, vol \overline{C})} \tag{C.14}$$

and

$$\phi_G = \min_{C \subset V} \phi_G(C) \tag{C.15}$$

is called the conductance of the graph G (or Cheeger constant). □

The following relation holds between the conductance of an undirected graph and the Fiedler value of its normalised Laplacian:

$$\frac{1}{2}\phi_G^2 \leq \lambda_1 \leq 2\phi_G \tag{C.16}$$

This is the so-called Cheeger inequality [75]. It implies that if λ_1 is a small number, then the graph possesses also a cut with low conductance (sparsity coefficient).

Definition C.2.2 *Normalised Laplacian of the graph G is defined as the symmetric matrix*

$$\mathfrak{L} = D^{-1/2}LD^{-1/2} = \mathbb{I} - D^{-1/2}SD^{-1/2} \tag{C.17}$$

□

While the equation (C.11) holds for any nonzero vector $\mathbf{x}$ in case of a combinatorial Laplacian, the following analogous equation holds for a normalised Laplacian

$$\mathbf{x}^{\mathsf{T}}\mathfrak{L}\mathbf{x} = \sum_{(v_i,v_j)\in E} s_{ij}\left(\frac{\mathbf{x}_i}{\sqrt{\mathsf{d}_i}} - \frac{\mathbf{x}_j}{\sqrt{\mathsf{d}_j}}\right)^2 \tag{C.18}$$

C.2.1.2 Pseudo-inverse of the Laplacian

It has been stated in the preceding section that L is a singular matrix. It is, however, possible to compute the pseudo-inverse of the Laplacian, (called also Moore-Penrose inverse), that is, a matrix $L^{\dagger} = [l^{\dagger}_{ij}]$ fulfilling the following conditions

$$\begin{aligned} LL^{\dagger}L = L, &\quad L^{+}LL^{\dagger} = L^{\dagger} \\ (LL^{\dagger})^{\mathsf{T}} = LL^{\dagger}, &\ (L^{\dagger}L)^{\mathsf{T}} = L^{\dagger}L \end{aligned} \tag{C.19}$$

The pseudo-inverse $L^{\dagger}$ is characterised by properties, [351], [150]:

(a) $L^{\dagger}$ is a symmetric and positive semi-definite matrix,
(b) Let $(\lambda_i, \mathbf{u}_i)$ be an eigenpair of a Laplacian. If $\lambda_i \neq 0$ then $(\lambda_i^{-1}, \mathbf{u}_i)$ is an eigenpair of its pseudo-inverse. Otherwise, $(\lambda_i, \mathbf{u}_i)$ is also an eigenpair of the matrix $L^{\dagger}$.
(c) $(\mathbf{e}^{\mathsf{T}}L^{\dagger})^{\mathsf{T}} = L^{\dagger}\mathbf{e} = 0$.
(d) $L^{\dagger}$ is a Gram matrix.

The property (b) means that $L^{\dagger}$ can be represented as the product

$$L^{\dagger} = U\Lambda^{\dagger}U^{\mathsf{T}} \tag{C.20}$$

where $U = (\mathbf{u}, \dots, \mathbf{u}_m)$ is a matrix with columns being eigenvectors of the Laplacian, and $\Lambda^{\dagger}$ is a diagonal matrix having elements

$$\lambda^{\dagger}_{ii} = \begin{cases} 1/\lambda_i \text{ if } \lambda_i > 0 \\ 0 \qquad \text{otherwise} \end{cases} \tag{C.21}$$

where λ_i means an eigenvalue of the Laplacian. Equation (C.20) implies that we can compute the elements of the pseudo-inverse from the eigenvalues and eigenvectors of a Laplacian as follows:

$$l^\dagger_{ij} = \sum_{k=1}^{m} u_{ik} u_{jk} \lambda^\dagger_k \tag{C.22}$$

Property (d) means that, according to the definition B.2.3, there exists a matrix $Y = (\mathbf{y}_1, \dots, \mathbf{y}_m)$ such that $l^\dagger_{ij} = \mathbf{y}_i^{\mathsf{T}} \mathbf{y}_j$. Hence, we can treat elements of the matrix $L^\dagger$ as degrees of similarity between objects i and j, see e.g. [130].

Another way to determine the pseudo-inverse is based on the formula

$$L^\dagger = \big(L - \frac{1}{m}\mathbf{e}\mathbf{e}^{\mathsf{T}}\big)^{-1} + \frac{1}{m}\mathbf{e}\mathbf{e}^{\mathsf{T}} \tag{C.23}$$

An efficient method of using this formula in the case of large graphs has been presented in section 4.3 of the paper [130].

Definition C.2.3 *Let $G = (V, E)$ be a connected and undirected graph and let $\mathbf{L}^\dagger$ be a pseudo-inverse of its Laplacian. The resistance distance c_{ij} between the nodes v_i, v_j is equal*

$$c_{ij} = (vol V)(l^\dagger_{ii} - 2l^\dagger_{ij} + l^\dagger_{jj}) = (vol V)(\mathbf{e}_i - \mathbf{e}_j)^T L^\dagger (\mathbf{e}_i - \mathbf{e}_j) \tag{C.24}$$

where $\mathbf{e}_i$ denotes i^{th} column of a unit matrix. □

It turns out that $\sqrt{c_{ij}}$ possesses the properties of Euclidean distance. Hence, nodes of the graph G can be assigned points $z_i \in \mathbb{R}^n$, $i = 1, \dots, m$, in such a way that distances between them are equal to resistance distances. Precisely speaking, the coordinates of these points are identical with rows of the matrix $U(\Lambda^\dagger)^{1/2}$. Thus, we have $\mathbf{z}_i^{\mathsf{T}} \mathbf{z}_j = \mathbf{e}_i^{\mathsf{T}} L^\dagger \mathbf{e}_j$ as well as $c_{ij} = (\mathrm{vol}\, V)\|z_i - z_j\|^2$. More information on the topic can be found in section 6 of the paper [351] and in the references cited therein. We present in section D.1.2 other examples of application of pseudo-inversion.

C.2.1.3 Laplacian of a directed graph

Laplacian of a directed graph is a more subtle object than a Laplacian of an undirected graph. The proposal presented here stems from Chung [76].

Let $G = (V, E)$ be a directed graph of the neighbourhood matrix S. Its elements s_{ij} are positive, if v_j is a direct neighbour of the node v_i, tzn. $v_i \to v_j$, and $s_{ij} = 0$ otherwise. Now we have to distinguish between in-degree d_j^{in} and out-degree d_j^{out}; they are defined by equations (C.7) and (C.8)

If $\mathsf{d}_j^{out} > 0$ holds for each node, then we can construct the Markov chain over the transition matrix

$$P = D^{-1} S \tag{C.25}$$

where D is a diagonal matrix with elements $d_j = \mathsf{d}_j^{out}$. Let $\boldsymbol{\pi}$ be a stationary distribution of this chain, i.e. $\boldsymbol{\pi}$ is a row vector fulfilling the equation $\boldsymbol{\pi} = \boldsymbol{\pi} P$.

Let us recall that the condition of existence and uniqueness of such a solution is that G be a strongly connected and aperiodic graph. Chung [77] proposes the following construction [4]:

$$\widetilde{L} = \frac{\Pi P + P^{\mathsf{T}} P}{2} \tag{C.26}$$

It is an analogue of the combinatorial Laplacian, defined for the undirected graph. A normalised variant is of the form, see [76], [391]:

$$\mathcal{L} = \mathbb{I} - \frac{1}{2}\left(\Pi^{1/2} P \Pi^{-1/2} + \Pi^{-1/2} P^{\mathsf{T}} \Pi^{1/2}\right) \tag{C.27}$$

where $\Pi = \mathtt{diag}(\boldsymbol{\pi})$.

For a Laplacian defined in this way, the methods elaborated for undirected graph are applied.

C.2.2 Green functions

Basic results characterising the Green function for undirected graphs were obtained by Chung and Yau[5]. We present in this section those related basic formulas that are exploited in this book. We make use of the formalism of [288].

Formally, if $\mathfrak{L}$ is a normalised Laplacian, corresponding to the similarity matrix S, and $\Delta = D^{-1/2}\mathfrak{L}D^{1/2}$ is a discrete Laplace operator, then a discrete Greeen function is understood as the left inverse of the operator Δ, i.e. matrix G such that $G\Delta = \mathbb{I}$.

Let $\lambda_1 \leq \lambda_2, \leq \dots le\lambda_m$ be increasingly ordered eigenvalues of the combinatorial Laplacian L and let $\mathbf{v}_i$ mean Laplacian's eigenvector corresponding to the i^{th} eigenvalue ($i = 1, \dots, m$). Similarly, let $\lambda'_1 \leq \lambda'_2, \leq \dots \leq \lambda'_m$ denote ordered eigenvalues of the normalised Laplacian $\mathfrak{L}$, and let $\mathbf{w}_i$ denote the eigenvector corresponding to the i^{th} eigenvalue of the matrix $\mathfrak{L}$.

Elements $\overline{g}_{ij}$ of Green function, corresponding to the combinatorial Laplacian L, may be determined from the equation

$$\overline{g}_{ij} = \sum_{u=2}^{m} \frac{1}{\lambda_u} v_{iu} v_{ju} \tag{C.28}$$

and elements $\mathfrak{g}_{ij}$ of the Green function, corresponding to the normalised Laplacian $\mathfrak{L}$, are determined from the equation

$$\mathfrak{g}_{ij} = \sum_{u=2}^{m} \frac{1}{\lambda'_u} w_{iu} w_{ju} \tag{C.29}$$

[4] Many interesting remarks are contained in the report of D.F. Gleich, Hierarchical directed spectral graph partitioning. *Information Networks, Stanford University, Final Project*, 2005, URL: `http://www.cs.purdue.edu/homes/dgleich/publications/Gleich2005-hierarchicaldirectedspectral.pdf`.

[5] F. Chung and S.-T. Yau, "Discrete Green's functions", *J. Combinatorial Theory Ser.*, pp. 191-214, 2000.

Symbol v_{iu} (resp. w_{iu}) denotes the i^{th} element of the eigenvector $\mathbf{v}_u$ (resp. $\mathbf{w}_u$).

A comparison of these formulas with equation (C.22) allows us to conclude that the Green matrix $\overline{G} = [\overline{g}_{ij}]$ is a pseudo-inverse of the combinatorial Laplacian L, and $\mathfrak{G} = [\mathfrak{g}_{ij}]$ is a pseudo-inverse of the normalised Laplacian $\mathfrak{L}$. Furthermore, one can check that

$$\begin{aligned}\overline{G}L = L\overline{G} = \mathbb{I} - \mathbf{v}_1\mathbf{v}_1^{\mathsf{T}}\\ \mathfrak{G}\mathfrak{L} = \mathfrak{L}\mathfrak{G} = \mathbb{I} - \mathbf{w}_1\mathbf{w}_1^{\mathsf{T}}\end{aligned}$$

D

Random walk on a graph

A random walk over an undirected graph is a classic example of a (discrete) Markov chain. In this Appendix we are dealing with some specific features of such a random walk on undirected graphs, which apply to the issues of clustering of objects. A reader who is interested in the relationship between random walk and solving heat conduction equations or random walk and resistance networks is recommended to consult papers [239], [110], [227].

D.1 Random walk on undirected graphs

D.1.1 Basic facts

Let $G = (V, E)$ be an undirected graph with m nodes. The probability of transition from the node j to the neighbouring node i is equal $1/\mathsf{d}_j$, i.e. each node belonging to the set of neighbours $N(j)$ of the node j is selected with same probability. Such a walk represents the indifference of the walker: when he is at the node j, he chooses with equal probability one of d_j edges leaving this node. If G is an undirected graph with neighbourhood matrix A then

$$P = AD^{-1} \tag{D.1}$$

is the transfer matrix characterising the random walk process. D denotes the matrix of node degrees, $D = \text{diag}(\mathsf{d}_1, \ldots, \mathsf{d}_m)$.

Remark D.1.1 *Matrix P from equation (D.1) is a column-stochastic matrix, meaning that the sum of elements in each column is equal 1. If we denote the initial probability distribution over the set of nodes with $\mathbf{p}^0$, then this distribution takes on the form $\mathbf{p}^1 = P\mathbf{p}^0$ after the first step. The number p_{ij} is called probability of passage (or transfer) from the node j to the node i (the index sequence may seem a bit counter-intuitive here).*

In the Markov chain theory (see, e.g., [188]), the column stochastic matrix P is frequently replaced by the row stochastic matrix $\widetilde{P} = P^T = D^{-1}A$. In this case, the element $\widetilde{p}_{ij}$ represents the probability of transition from the node i to the node j. Hence, if w^0 is a row *vector, representing the initial probability distribution over the set of nodes, then after one step this distribution will be of the form $w^1 = w^0\widetilde{P}$. The price to be paid for the use of the transition matrix with a more natural interpretation of its elements is the necessity to use row* *vectors and to apply left side multiplication of the row vectors with the transition matrix.*

Though both representations yield the very same results, we will use the representation (D.1) as the right-hand sided multiplication of the column vector with a matrix is more efficient that left-sided multiplication of a row vector by a matrix[1]*.* □

If G is a connected graph, then the Perron-Frobenius theorem implies that P has at least one (right) vector $\boldsymbol{\pi}$ corresponding to the principal eigenvalue $\lambda_{max} = 1$, and all the components of this vector are positive. Further, if G is aperiodic, then there exists exactly one such vector. *A random walk on a graph G is connected if and only if G is connected, and is aperiodic if and only if G is not bipartite.* The probability distribution represented by the (normalised[2]) vector $\boldsymbol{\pi}$ is called stationary distribution. The above implies

$$P\boldsymbol{\pi} = \boldsymbol{\pi} \tag{D.2}$$

An element π_i of this vector can be treated as the inverse of the expected number of steps necessary to return to the node $v_i \in V$ on a random walk according to the distribution P when starting at this node.

We summarise below some important facts about the properties of such a random walk.

Lemma D.1.1 *Let $G(V, E)$ be a connected and aperiodic undirected graph and let $\mathsf{d} = (\mathsf{d}_1, \dots, \mathsf{d}_m)^T$ denote the vector of degrees. Then the vector*

$$\boldsymbol{\pi} = \frac{\mathsf{d}}{\sum_i \mathsf{d}_i} = \frac{\mathsf{d}}{vol V} \tag{D.3}$$

represents a stationary distribution of a Markov chain with transfer matrix P.

Proof: *$D^{-1}\mathsf{d} = \mathbf{e}$, hence*

$$P\mathsf{d} = AD^{-1}\mathsf{d} = A\mathbf{e} = \mathsf{d}$$

i.e. d is a right eigenvector of the matrix P (note that the last equality is a consequence of the symmetry of the matrix A). By normalising this vector we obtain a stochastic vector $\boldsymbol{\pi}$ of the abovementioned form. □

Lemma D.1.2 *A Markov chain with the transfer matrix (D.1) is a reversible chain, i.e..*

$$p_{ij}\pi_j = p_{ji}\pi_i$$

Proof *results from the symmetry of the matrix A and the fact that $p_{ij} = a_{ij}/\mathsf{d}_j$, and $\pi_j = \mathsf{d}_j/vol X$.* □

[1] See G. Gundersen and T. Steihaug. Data structures in Java for matrix computations. In: *Proc. Norwegian Informatics Conf., NIK'2002*, pp. 97-108, Kongsberg, Norway, 2002. URL: `http://www.nik.no/2002/Gundersen.pdf`.

[2] Here, normalisation means that the sum of components of the vector is equal 1

To ensure aperiodicity, the so-called lazy random walk is introduced, represented by the transfer matrix

$$\widehat{P} = \alpha\mathbb{I} + (1-\alpha)P, \quad 0 < \alpha < 1 \tag{D.4}$$

that is, with probability α the walker does not change the location (hence the name "lazy random walk"), and with probability $(1-\alpha)$ moves to a randomly selected neighbour of the current node. Usually, $\alpha = 1/2$ is assumed. Such a solution is suggested both by Chung [77] and Zhou and Schölkopf [392].

Lemma D.1.3 *Matrices P and $\widehat{P}$ have identical eigenvectors.*

Proof: *Let $(\lambda, \mathbf{w})$ be an eigenpair of the matrix P, i.e. $P\mathbf{w} = \lambda\mathbf{w}$. Then*

$$\widehat{P}\mathbf{w} = \alpha\mathbf{w} + (1-\alpha)\lambda\mathbf{w} = [\alpha + (1-\alpha)\lambda]\mathbf{w} = \widehat{\lambda}\mathbf{w}$$

hence, $(\widehat{\lambda}, \mathbf{w})$ is an eigenpair of the matrix $\widehat{P}$. □

The interval $[-1, 1]$, containing all eigenvalues of the matrix P, is now transformed into the interval $[2\alpha - 1, 1]$, containing eigenvalues of the matrix $\widehat{P}$. In particular, if $\alpha = 1/2$, then $\widehat{\lambda} = (1+\lambda)/2$, that is, the eigenvalues of the matrix $\widehat{P}$ belong to the set $[0, 1]$. This property is a justification for the choice of $\alpha = 1/2$. The above lemma implies also that, independently of the value of α, the vector $\boldsymbol{\pi}$, defined by the formula (D.3), is a stationary distribution of the Markov chain with transfer matrix $\widehat{P}$.

Matrices P and $\widehat{P}$ are not symmetric. To derive related symmetric matrices, we apply the trick known from section, 5.2.4 consisting in multiplication by special matrices:

$$\mathcal{P} = D^{-1/2}PD^{1/2} = D^{-1/2}AD^{-1/2} \tag{D.5}$$

and

$$\widehat{\mathcal{P}} = D^{-1/2}\widehat{P}D^{1/2} = \alpha\mathbb{I} + (1-\alpha)D^{-1/2}AD^{-1/2} \tag{D.6}$$

We see in equation (D.5) the complement of the normalised Laplacian (5.51) from section 5.2.4.

Note also that both matrices can be obtained by an analogous trick, applied to the symmetric row-stochastic matrix $\widetilde{P}$, i.e.

$$\begin{aligned}
\mathcal{P} &= D^{1/2}\widetilde{P}D^{-1/2} \\
\widehat{\mathcal{P}} &= D^{1/2}\Big(\alpha\mathbb{I} + (1-\alpha)\widetilde{P}\Big)D^{-1/2} = \alpha\mathbb{I} + (1-\alpha)D^{-1/2}AD^{-1/2}
\end{aligned}$$

The following simple lemma is worth noting:

Lemma D.1.4 *If $(\lambda, \mathbf{w})$ is an eigenpair of a symmetric matrix, $\mathcal{P}$ then $(\lambda, D^{1/2}\mathbf{w})$ is an eigenpair of the matrix P.*

Proof: *Since $(\lambda, \mathbf{w})$ is an eigenpair of the matrix $\mathcal{P}$, we have*

$$\lambda \mathbf{w} = \mathcal{P}\mathbf{w} = D^{-1/2}PD^{1/2}\mathbf{w} = D^{-1/2}P\mathbf{v}$$

where $\mathbf{v} = D^{1/2}\mathbf{w}$. *By multiplying both sides of the above equality by* $D^{1/2}$, *we obtain* $\lambda D^{1/2}\mathbf{w} = P\mathbf{v}$, *that is,* $\lambda\mathbf{v} = P\mathbf{v}$. □

Lemma D.1.5 *Let* $G = (V, E)$ *be an undirected graph, described by the neighbourhood matrix* A, *and let* $P = AD^{-1}$ *be a transfer matrix. If the pair* $(\lambda, \mathbf{w})$ *is the solution of the eigen problem*

$$P\mathbf{w} = \lambda\mathbf{w}$$

then the pair $(1-\lambda, \mathbf{w})$ *is the solution of the generalised eigen problem (5.45).* □

An elementary proof of this lemma was presented in section 5.2.4 while justifying the algorithm 5.3.

D.1.2 Characteristics of random walk

The times, required to reach some states are among the important characterisations of a random walk. We present below their definitions and related computational methods.

D.1.2.1 Mean access time

The mean hitting time[3] (*hitting time*), h_{ij}, is understood as the expected number of steps after which a random walker starting his journey at node j will reach the node i. This time can be computed in several ways.

(a) A naive method consists in transforming the Markov chain with transfer matrix P, describing a random walk over a connected graph, into a chain with a single absorbing state i, characterised by a matrix $\widehat{P}$. If we assign numerical identifiers to chain states in such a way that the state i gets the identifier 1 then the matrix $\widehat{P}$ is of the form[4]

$$\widehat{P} = \begin{bmatrix} 1 & R \\ \mathbf{0} & T \end{bmatrix}$$

The average number of steps needed to reach the absorbing state from any state is equal to the components of the vector $\mathbf{t} = (\mathbb{I} - T^{\mathsf{T}})^{-1}\mathbf{e}$.

(b) Another method was presented in [130]. Let $L^{\dagger} = [l^{\dagger}_{ij}]$ be the pseudo-inverse of the Laplacian, see section C.2.1.2. The expected number of steps h_{ij} required to reach state i from state j is equal

$$h_{ij} = \sum_{k=1}^{m} \left(l^{\dagger}_{jk} - l^{\dagger}_{ji} - l^{\dagger}_{ik} + l^{\dagger}_{ii}\right)\mathsf{d}_k \tag{D.7}$$

[3] It is called in the literature also *first passage time*, *mean escape time* or *mean reaching time*.

[4] Please remember that P is a column-stochastic matrix !

(c) Lovász in [239] provides a further method. Let $\widetilde{P} = \mathbb{I} - \mathfrak{L} = D^{-1/2}AD^{-1/2}$ and let $(\lambda_i, \mathbf{u}_i)$ be eigen pairs of this matrix, ordered by decreasing eigenvalues $1 = \lambda_1 > \lambda_2 \geq \cdots \geq \lambda_m$. Then, see [239, Thm. 3.1]

$$h_{ij} = 2\mathfrak{m} \sum_{k=2}^{m} \frac{1}{1-\lambda_k} \Big(\frac{u_{ik}^2}{\mathsf{d}_i} - \frac{u_{ik}u_{jk}}{\sqrt{\mathsf{d}_i \mathsf{d}_j}} \Big) \tag{D.8}$$

D.1.2.2 Commute time

The *commute time* , c_{ij}, is the average number of steps, after which a random walker starting at node i reaches node j, and from there he returns to node i, i.e. $c_{ij} = h_{ij} + h_{ji}$. One can compute it directly from the definition or use the equation below, see [130, Appendix C]

$$c_{ij} = 2\mathfrak{m}(l_{ii}^{+} + l_{jj}^{+} - 2l_{ij}^{+}) \tag{D.9}$$

This equality can be rewritten as

$$c_{ij} = 2\mathfrak{m}(\mathbf{e}_i - \mathbf{e}_j)^{\mathtt{T}} L^{\dagger} (\mathbf{e}_i - \mathbf{e}_j) \tag{D.10}$$

where $\mathbf{e}_i$ means the i^{th} column of unit matrix.

Using notation introduced in section C.2.2, the commute time can be equivalently computed from the formulas[5]

$$\begin{aligned} c_{ij} &= 2\mathfrak{m} \sum_{u=2}^{m} \frac{1}{\lambda_u}' \left(\frac{w_{iu}}{\sqrt{d_i}} - \frac{w_{ju}}{\sqrt{d_j}} \right)^2 \\ &= 2\mathfrak{m} \sum_{u=2}^{m} \frac{1}{\lambda_u} \Big(v_{iu} - v_{ju} \Big)^2 \end{aligned} \tag{D.11}$$

If we treat the nodes of the graph in the space of nodes, that is, we associate i^{th} node with the vector $\mathbf{e}_i$ and apply the equation (2.2) from page 26 then we can state that $\sqrt{c_{ij}}$ is a distance in the space spanned over the set of vectors $\mathbf{e}_i$, $i = 1, \ldots, m$.

In fact, the commute time c_{ij} is a distance. It is related to the so-called resistance distance r_{ij} via the formula [68]

$$c_{ij} = 2\mathfrak{m} r_{ij} \tag{D.12}$$

Bapat, Gutman and Xiao[6] provided a simple formula allowing to compute the resistance distance:

[5] They are correct under the assumption that the similarity matrix S represents a connected graph. Only in such a case the corresponding Laplacian has exactly one eigenvalue equal to zero.

[6] R.B. Bapat, I. Gutman, W. Xiao. A simple method for computing resistance distance. *Z. Naturforsch.*, **58**: 2003, pp. 494–498.

$$r_{ij} = \frac{\det L(i,j)}{\det L(i)} \tag{D.13}$$

where $L(i)$ means the matrix L (Laplacian), from which i^{th} row and i^{th} column have been removed, and $L(i,j)$ is the Laplacian, from which rows and columns with numbers i and j have been removed. Let us stress that $L(i)$ is constant for each i, so that the denominator needs to be computed only once.

D.1.2.3 Coverage time

Coverage time, c_i, is the expected number of steps required for a random walker who starts at node i to visit all the nodes. The number $C(G) = \max_i c_i$ is called the coverage time of the graph G. One can demonstrate that $C(G) \leq \mathfrak{m}(m-1)$.

D.1.2.4 Mixing time

Mixing speed, μ, shows how quickly the random walk approaches the stationary distribution

$$\mu = \limsup_{t\to\infty} \max_{i,j} |p_{ij}(t) - \pi_j|^{1/t} \tag{D.14}$$

Here, $p_{ij}(t)$ means the probability of passing from node i to j in t steps.

The inverse of mixing speed is called mixing time. It tells after how many steps the matrix P turns into the stationary matrix P^∞, that is - becomes a matrix of the form $\mathbf{e}\boldsymbol{\pi}^\mathsf{T}$. An approximate value of this time is $\tau = (\log m)/(1-\lambda_2)$, where λ_2 is the second principal eigenvalue of the matrix P. However, its exact value depends on the understanding of the proximity between the matrices P^t and P^∞, see [239].

D.2 Random walk on directed graphs

Random walk in a directed graph $G = (V, E)$, endowed with the neighbourhood matrix W, is described by the probabilities

$$p_{ij} = \begin{cases} w_{ij}/\mathsf{d}_i^+ & \text{if } (v_i, v_j) \in E \\ 0 & \text{otherwise} \end{cases} \tag{D.15}$$

where d_i^+ is the out-degree of the node i, that is, the number of arcs leaving this node.

If G is a strongly connected graph, then there exists a unique stationary distribution $\boldsymbol{\pi}$ with positive components, such that the condition $\pi_j = \sum_{v_i \to v_j} \pi_i p_{ij}$ holds. Regrettably, the distribution $\boldsymbol{\pi}$ does not have an analytical form in this case, contrary to undirected graphs. We find it by applying e.g. the power method, that is - by iterating the equation $\boldsymbol{\pi}^{(k+1)} = P\boldsymbol{\pi}^{(k)}$ till $\|\boldsymbol{\pi}^{(k+1)} - \boldsymbol{\pi}^{(k)}\| \leq \epsilon$, where ϵ is a predefined precision.

The matrix P needs to be modified if G is not a strongly connected graph. Zhou Huang and Schölkopf propose the following construction of the random walk with teleportation [391]. If the random walker reached a node u with a positive out-degree d_u^+, then in the next step he moves to: (a) a (uniformly) randomly selected node $v \neq u$ or (b) one of the nodes pointed to by the arcs $u \to v$. Decision (a) is made with probability $1 - \eta$, and decision (b) – with probability $\eta > 0$. Otherwise, if $\mathsf{d}_u^+ = 0$, then the random walker jumps to any node $v \in X \backslash \{u\}$. Hence, the elements of the transfer matrix are of the form

$$p_{ij}^{tele} = \begin{cases} \eta \dfrac{w_{ij}}{\mathsf{d}_i^+} + \dfrac{1-\eta}{m-1} \text{ if } \mathsf{d}_i^+ > 0 \text{ and } i \neq j \\ \eta \dfrac{w_{ij}}{\mathsf{d}_i^+} \text{ if } \mathsf{d}_i^+ > 0 \text{ and } i = j \\ 1/(m-1) \text{ if } \mathsf{d}_i^+ = 0 \text{ and } i \neq j \\ 0 \text{ otherwise} \end{cases} \tag{D.16}$$

Another variant of random walk with teleportation, proposed in [274], consists in construction of the transfer matrix of the form

$$P^{tele} = \xi P + \frac{1-\xi}{m} \mathbf{e}\mathbf{e}^{\mathsf{T}} \tag{D.17}$$

given that P is an aperiodic transfer matrix.

E

Personalized PageRank vector

In case of undirected graphs, the PageRank vector corresponds to a stationary distribution of random the walk, described by a transfer matrix of a special form, see the remark E.1.1. We are interested in a special case of such a random walk, in which the random walker is located in a particular initial node. Knowledge of such stationary distributions allows to construct a stationary distribution for any stochastic initial vector. We discuss below some specific features of such a random walk.

E.1 Basic notions and interdependences

Let $G = (V, E)$ be an undirected and connected graph, and $P = AD^{-1}$ be a (column) stochastic transition matrix, describing random walk in this graph. As before, A is the neighbourhood matrix representing links in graph G, and D is the diagonal degree matrix. The PageRank vector is defined, as proposed by Page and Brin, in [274], as the vector $\rho(\mathbf{s}, \beta)$ being the solution to the equation

$$\rho(\mathbf{s}, \beta) = \beta\mathbf{s} + (1-\beta)P\rho(\mathbf{s}, \beta) \tag{E.1}$$

where $\beta \in (0, 1]$ is the so-called damping factor, and $\mathbf{s}$ is the starting vector. In the original formulation, $\mathbf{s}$ is a vector with all elements equal to $1/|V|$. If only some elements of this vector are positive, while the other ones are equal zero, then we talk about the *personalized* PageRank vector. The set $supp(\mathbf{s}) = \{v \in V : \mathbf{s}(v) > 0\}$ indicates the range of personalization. Subsequently, we will denote a personalised PageRank vector with the symbol $\mathfrak{p}(\mathbf{s}, \beta)$.

Lemma E.1.1 *PageRank vector possesses the following properties:*
(a) $\rho(\mathbf{s}, 1) = \mathbf{s}$,
(b) If π is a stationary distribution with components of the form (D.3) then $\rho(\pi, \beta) = \pi$ holds for any value of the parameter $\beta \in (0, 1]$,
(c) $\|\rho(\mathbf{s}, \beta)\|_1 = 1$ *if only* $\|\mathbf{s}\|_1 = 1$.

□

Remark E.1.1 *Please pay attention that in the original formulation of Page and Brin G is a directed graph, which requires a more careful transformation of the connection matrix into a stochastic matrix. Furthermore, in order to ensure that the corresponding transfer matrix has a single principal value, the authors modify the stochastic matrix P to the form $P' = \beta\mathbf{s}\mathbf{e}^T + (1-\beta)P$. If we denote* *with $\rho(\mathbf{s},\beta)$ the eigenvector, corresponding to the principal eigenvalue (equal 1) of the matrix P', we can easily check that, in fact, it fulfills the conditions of the equation (E.1), i.e.*

$$P'\rho(\mathbf{s},\beta) = \beta\mathbf{s}(\mathbf{e}^T\rho(\mathbf{s},\beta)) + (1-\beta)P\rho(\mathbf{s},\beta)$$

because $\mathbf{e}^T\rho(\mathbf{s},\beta) = 1$.

As indicated in the remark D.1.1 on page 254, many authors assume that ρ and $\mathbf{s}$ are row *vectors. In such a case the equation (E.1) is of the form*

$$\rho(\mathbf{s},\beta) = \beta\mathbf{s} + (1-\beta)\rho(\mathbf{s},\beta)P^T$$

i.e. the matrix P^T is defined as the product $D^{-1}A$. □

Remark E.1.2 *Let us stress that the symbols $\rho(\mathbf{s},\beta)$ and $\mathfrak{p}(\mathbf{s},\beta)$ denote the same solution of the equation system (E.1). We introduce them solely for the convenience of the Reader. When we use the symbol $\rho(\mathbf{s},\beta)$, we have in mind the global PageRank vector, that is, a solution obtained for the* positive *vector* $\mathbf{s}$, *whereas the symbol $\mathfrak{p}(\mathbf{s},\beta)$ is intended to mean the personalised PageRank vector that is the solution of equation system (E.1) for the* nonnegative *starting vector* $\mathbf{s}$. □

The transfer matrix P, which appears in equation (E.1), was replaced in the paper [13] by the *lazy* random walk matrix of the form

$$\widehat{P} = \frac{1}{2}(I + AD^{-1}) \tag{E.2}$$

In such a case, the personalised PageRank vector is a vector $\mathfrak{p}(\mathbf{s},\alpha)$, for which the equation below holds:

$$\mathfrak{p}(\mathbf{s},\alpha) = \alpha\mathbf{s} + (1-\alpha)\widehat{P}\mathfrak{p}(\mathbf{s},\alpha) = \alpha\mathbf{s} + (1-\alpha)\frac{1}{2}(\mathbb{I}+P)\mathfrak{p}(\mathbf{s},\alpha) \tag{E.3}$$

Lemma E.1.2 *If $\mathfrak{p}(\mathbf{s},\alpha;\widehat{P})$ is a solution of the equation system (E.3), and $\mathfrak{p}(\mathbf{s},\beta;P)$ is a solution of the system (E.1), and if $\beta = 2\alpha/(1+\alpha)$, then $\mathfrak{p}(\mathbf{s},\alpha;\widehat{P}) = \mathfrak{p}(\mathbf{s},\beta;P)$.*

Proof: *By transforming the equation (E.3), we obtain*

$$\frac{1+\alpha}{2}\mathfrak{p}(\mathbf{s},\alpha;\widehat{P}) = \alpha\mathbf{s} + \frac{(1-\alpha)}{2}P\mathfrak{p}(\mathbf{s},\alpha;\widehat{P})$$

By multiplying both sides of the above equation by $2/(1+\alpha)$ and noticing that $(1-\alpha)/(1+\alpha) = 1-\beta$, where $\beta = 2\alpha/(1+\alpha)$, we obtain the thesis. □

The above lemma demonstrates that the equations (E.1) and (E.3) possess identical solutions if we apply in both cases the same starting vector and the damping factors satisfy the condition $\beta = 2\alpha/(1+\alpha)$, and matrices P and $\widehat{P}$ correspond to one another, according to equation (E.2). Furthermore, equations

(E.1) and (E.3), defining $\mathfrak{p}(\mathbf{s},\beta)$ and $\mathfrak{p}(\mathbf{s},\alpha)$, have exactly the same formal form. Hence, algebraic considerations, as well as algorithms derived for $\mathfrak{p}(\mathbf{s},\beta)$ and β, P, can be mapped by simple symbol substitution into $\mathfrak{p}(\mathbf{s},\alpha)$ and $\alpha, \widehat{P}$ and vice versa. Therefore, we will limit ourselves to seeking the distributions of $\mathfrak{p}$ for $\alpha, \widehat{P}$ only.

Lemma E.1.3 *A PageRank vector, personalised with respect to the starting vector* $\mathbf{s}$*, is a sum of the geometric series:*

$$\mathfrak{p}(\mathbf{s},\alpha) = \alpha \sum_{t=0}^{\infty} (1-\alpha)^t \widehat{P}^t \mathbf{s} = \alpha \mathbf{s} + \alpha \sum_{t=1}^{\infty} (1-\alpha)^t \widehat{P}^t \mathbf{s} \tag{E.4}$$

Proof: *Let us rewrite the equation (E.3) in the form*

$$[\mathbb{I} - (1-\alpha)\widehat{P}]\mathfrak{p}(\mathbf{s},\alpha) = \alpha\mathbf{s}$$

This implies that $\mathfrak{p}(\mathbf{s},\alpha) = \alpha[\mathbb{I} - (1-\alpha)\widehat{P}]^{-1}\mathbf{s}$ *if there exists the matrix* $[\mathbb{I} - (1-\alpha)\widehat{P}]^{-1}$*. Theorem 1.7 in [188] leads to the conclusion that if* X *is a square matrix such that* $X^n \to 0$ *when* $n \to \infty$*, then the matrix* $(\mathbb{I} - X)$ *possesses an inverse matrix of the form*

$$(\mathbb{I} - X)^{-1} = \sum_{t=0}^{\infty} X^t$$

Substituting the matrix $(1-\alpha)\widehat{P}$ *for* X *we can check that the condition of the theorem is fulfilled, hence the conclusion.* □

Note that in the proof of the above lemma we represented the personalised PageRank vector in the form $\mathfrak{p}(\mathbf{s},\alpha) = \alpha[\mathbb{I} - (1-\alpha)\widehat{P}]^{-1}\mathbf{s}$.

By substituting

$$R_\alpha = \alpha\mathbb{I} + \alpha \sum_{t=1}^{\infty} (1-\alpha)^t \widehat{P}^t \tag{E.5}$$

we can rewrite the equation (E.4) in the form

$$\mathfrak{p}(\mathbf{s},\alpha) = R_\alpha \mathbf{s} \tag{E.6}$$

This equation makes apparent that the presonalised PageRank vector is a linear transformation of the vector $\mathbf{s}$, and R_α is the matrix of this transformation. Two important properties of the PageRank vector can be derived from it.

Lemma E.1.4 *The PageRank vector possesses the following properties:*

(a) It is linear with respect to the starting vector, i.e. $\mathfrak{p}(\mathbf{s}_1 + \mathbf{s}_2, \alpha) = \mathfrak{p}(\mathbf{s}_1, \alpha) + \mathfrak{p}(\mathbf{s}_2, \alpha)$ *for any vectors* $\mathbf{s}_1, \mathbf{s}_2$*. This, in turn, means that the standard PageRank vector* $\rho(\mathbf{s},\alpha)$ *is a weighted average of personalised PageRank vectors* $\mathfrak{p}(\chi_v, \alpha)$*,* $v \in V$*. Here, the symbol* χ_v *denotes the characteristic function of the set* $S = \{v\}$*.*

(b) The operator $\mathfrak{p}(\mathbf{s},\alpha)$ commutes with the matrix $\widehat{P}$, i.e. $\widehat{P}\mathfrak{p}(\mathbf{s},\alpha) = \mathfrak{p}(\widehat{P}s,\alpha)$. The equality below is a consequence of this property

$$\mathfrak{p}(\mathbf{s},\alpha) = \alpha\mathbf{s} + (1-\alpha)\mathfrak{p}(\widehat{P}s,\alpha) \tag{E.7}$$

Proof: *The property (a) results directly from the formula $\mathfrak{p}(\mathbf{s},\alpha) = R_\alpha\mathbf{s}$. The property (b) results from a simple transformation*

$$\widehat{P}\mathfrak{p}(\mathbf{s},\alpha) = \alpha(\widehat{P}s) + (1-\alpha)\widehat{P}(\widehat{P}s) = \mathfrak{p}(\widehat{P}s,\alpha)$$

Taking this into account, we transform the definition of the PageRank vector into the form

$$\mathfrak{p}(\mathbf{s},\alpha) = \alpha\mathbf{s} + (1-\alpha)\widehat{P}\mathfrak{p}(\mathbf{s},\alpha) = \alpha\mathbf{s} + (1-\alpha)\mathfrak{p}(\widehat{P}s,\alpha)$$

□

E.2 Approximate algorithm of determining the personalized PageRank vector

The PageRank vector represents, in fact, a stationary distribution of the matrix $P' = (1-\beta)P + \beta\mathbf{s}\mathbf{e}^{\mathsf{T}}$, see remark E.1.1. If $G = (V,E)$ is a connected graph, then P is a regular matrix as in a finite number of steps one can pass from any node $u \in V$ to any other node $v \in V$. According to the theory of regular Markov chains, see e.g. [188], the stationary distribution of the matrix P' has the form

$$\mathfrak{p}_i(\mathbf{s},\alpha) = \frac{\Delta_i}{\sum_{j=1}^{m}\Delta_i}, i = 1,\ldots,m \tag{E.8}$$

where Δ_i is the algebraic complement of the i^{th} element, lying on the main diagonal in the matrix $\mathbb{I} - P'$, and $\mathfrak{p}_i(\mathbf{s},\alpha)$ is the i^{th} element of the personalised vector $\mathfrak{p}(\mathbf{s},\alpha)$.

Application of the above formula requires computation of m determinants of matrices of dimensions $(m-1)\times(m-1)$, which is quite expensive in practice. The equation (E.4) implies a conceptually simple method of determining the approximate value of the personalised PageRank vector. It is illustrated by the algorithm E.1.

A disadvantage of the algorithm is the necessity to perform multiple multiplications of a vector by a matrix. Taking into account that P is a sparse matrix, this algorithm can be applied to the middle sized graphs.

In case of large graphs, a simulation method, suggested in the paper [13], can prove to be much more convenient. The authors exploited there in an interesting way the idea of Berkhina, who proposed in [46] the following metaphor: let us place a portion of paint in the starting node v, which is spilled over the neighbouring nodes. In each step, a fraction α of the paint, present in a given node v dries out, half of the still wet paint remains in the node v, and the remaining part spills in equal proportions onto the neighbours of the node v. Let $\mathbf{p}^t$ be a vector with elements $p^t(v)$, representing the amount of dried paint

Algorithm E.1 Algorithm of determination of the approximation to the personalised PageRank vector on the basis of equation (E.4)

```
 1: *Input parameters* : Starting vector $\mathbf{s}$, damping coefficient $\alpha$, column-stochastic matrix $P$, precision $\epsilon$.
 2: $\mathfrak{p}_{old} = 0$, $\mathfrak{p}_{new} = 0$, $\mathbf{s}_1 = \mathbf{s}$.
 3: $\beta = 1$, $\alpha_1 = 1 - \alpha$
 4: $res = 1$
 5: **while** $(res > \epsilon)$ **do**
 6:     $\mathbf{s}_1 = P\mathbf{s}_1$
 7:     $\beta = \alpha_1\beta$
 8:     $\mathfrak{p}_{new} = \mathfrak{p}_{new} + \beta\mathbf{s}_1$
 9:     $res = \|\mathfrak{p}_{new} - \mathfrak{p}_{old}\|_1$
10:     $\mathfrak{p}_{old} = \mathfrak{p}_{new}$
11: **end while**
12: Return the vector $\mathfrak{p} = \alpha(\mathbf{s} + \mathfrak{p}_{new})$
```

at node $v \in V$ at the time point t, and let $\mathbf{r}^t$ be the vector having the elements $r^t(v)$ representing the amount of wet paint present at the time point t in the node v. The process of graph "colouring" is described by two equations

$$\begin{aligned} \mathbf{p}^{t+1} &= \mathbf{p}^t + \alpha\mathbf{r}^t \\ \mathbf{r}^{t+1} &= (1-\alpha)\widehat{P}\mathbf{r}^t \end{aligned} \tag{E.9}$$

where $\widehat{P}$ is a matrix of lazy random walk of the form (E.2), $\mathbf{p}^0 = \mathbf{0}$, and $\mathbf{r}^0 = \mathbf{e}_u$.

It is not difficult to see that the vector $\mathbf{p}^{t+1}$ assumes the form

$$\mathbf{p}^{t+1} = \alpha\sum_{k=0}^{t}\mathbf{r}^k = \alpha\sum_{k=0}^{t}(1-\alpha)^k\widehat{P}^k\mathbf{r}^0$$

By substituting $\mathbf{r}^0 = s$ and $t \to \infty$, we obtain the equation (E.4).

In the algorithm, presented in the paper [13], further simplifications were proposed, consisting in ignoring the time and in local perspective on the colouring process. The details of this approach are available in the cited paper. Let us only mention here that the essence of the algorithm is the creation of the so-called $\epsilon -$ *appproximation* of the vector $\mathfrak{p}(\mathbf{s}, \alpha)$, that is, a vector $\mathbf{p}$, for which the equation

$$\mathbf{p} + \mathfrak{p}(\mathbf{r}, \alpha) = \mathfrak{p}(\mathbf{s}, \alpha)$$

holds, where $\mathbf{s}$ is the starting vector, and $\mathbf{r}$ is a non-negative vector with components $r(v) < \epsilon\mathsf{d}(v)$, representing the amount of non-dried (wet) paint at nodes $v \in V$.

A further improvement to this algorithm was proposed by Chung and Zhao in [80]. Its time complexity was reduced to $O(\alpha\mathfrak{m}\log(1/\epsilon))$. The essence of this idea is illustrated by the pseudo-code E.2.

Let us underline that the method of choosing the node influences only the speed measured in terms of the number of iterations within the **while** loop. When lazy

Algorithm E.2 Fast algorithm of determining an ϵ-approximation of the personalised PageRank vector, [80]

```
1: Input parameters: Undirected graph $G = (V, E)$, initial vector (distribution) $\mathbf{s}$,
   coefficient $\alpha \in (0, 1)$, precision $\epsilon$.
2: Substitute $\mathbf{p} = 0$, $\mathbf{r} = \mathbf{s}$, $e = 1$.
3: while ($e > \epsilon$) do
4:   $e = e/2$
5:   $p' = 0$
6:   while ($\exists v \in V : r(v) \geq e\mathsf{d}(v)$) do
7:     Choose node $u$ such that $r(u) \geq \epsilon\mathsf{d}(u)$
8:     $p'(u) = p'(u) + \alpha r(u)$
9:     $r(v) = r(v) + \frac{1-\alpha}{\mathsf{d}(u)} r(u), \ \forall v \in N(u)$
10:    $r(u) = 0$
11:  end while
12:  $\mathbf{p} = \mathbf{p} + \mathbf{p}'$
13: end while
14: Return the vector $\mathbf{p}$
```

random walk is applied, about 50% more iterations are needed to find an approximation with required precision. The abovementioned results were obtained for the random walk with the random walk matrix P, teleportation coefficient $\alpha = 0.15$ and precision $\epsilon = 10^{-12}$. As comparisons show further on, the variant E.2 can be considered as the quickest one.

It turns out that in the graph G there are not many links between the nodes with high PageRank values and those with a low PageRank, [12]. More precisely, if we sort the graph nodes by the decreasing value $\mathfrak{p}(\mathbf{s}, \alpha)$, and the k^{th} node in this order is assigned a higher portion of probability than the node of rank $k(1 + \delta)$, then there exist few connections between nodes with ranks $\{1, \ldots, k\}$ and those with ranks from the set $\{k(1 + \delta) + 1, \ldots, |V|\}$.

F

Axiomatic systems for clustering

A considerable amount of research work has been devoted to understanding the essentials of clustering, as briefly discussed in Section 2.6.

A number of axiomatic frameworks[1] have been devised in order to capture the nature of the clustering process, the most often cited being probably the Kleinberg's system [206].

In general, the axiomatic frameworks of clustering address either:

- the required properties of clustering functions, or
- the required properties of the values of a clustering quality function, or
- the required properties of the relation between the qualities of different partitions (ordering of partitions for a particular set of objects and a given similarity/dissimilarity function).

We will now briefly overview Kleinberg's axioms and some work done around their implications.

F.1 Kleinberg's axioms

As a justification for his axiomatic system, Kleinberg [206] claims that a good clustering may only be a result of a reasonable method of clustering. His axioms are dealing with distance-based cluster analysis. He defines the clustering function as follows:

Definition F.1.1 *A function f is a* clustering function *if it takes as its argument a distance function d on the set S and returns a partition Γ of S. The sets in Γ will be called its* clusters. □

He postulated that some quite "natural" axioms need to be met, when we manipulate the distances between objects. These are:

Axiom F.1.1 *The clustering method should allow to obtain any clustering of the objects (so-called* richness property*). More formally, if $Range(f)$ denotes the set of all partitions Γ such that $f(d) = \Gamma$ for some distance function d then $Range(f)$ should be equal to the set of all partitions of S.*

[1] Axiomatic systems may be traced back to as early as 1973, when Wright proposed axioms of weighted clustering functions. This means that every domain object was attached a positive real-valued weight, that could be distributed among multiple clusters, like in later fuzzy systems. See: W.E. Wright. A formalization of cluster analysis. *Pattern Recognition*, **5**(3):273-282, 1973.

Axiom F.1.2 *The method should deliver clusterings invariant with respect to distance scale (so-called (scale)-*invariance property*). Formally: for any distance function d and any $\alpha > 0$, $f(d) = f(\alpha \cdot d)$ should hold.*

Axiom F.1.3 *The method should deliver the same clustering if we move objects closer to cluster centers to which they are assigned (so-called* consistency property*). Formally, for a partition Γ of S, and any two distance functions d and d' on S such that (a) for all $i, j \in S$ belonging to the same cluster of Γ , we have $d'(i,j) \leq d(i,j)$, and (b) for all $i, j \in S$ belonging to different clusters of Γ, we have $d'(i,j) \geq d(i,j)$ (we will say that d' is a Γ-transformation of d), the following must hold: if $f(d) = \Gamma$ then $f(d') = \Gamma$.*

F.1.1 Formal problems

Though the axioms may seem to be reasonable, Kleinberg demonstrated that they cannot be met all at once (but only pair-wise).

Theorem F.1.1 *No clustering function can have at the same time the properties of richness, invariance and consistency*

The axiom set is apparently not sound. The proof is achieved via contradiction. For example, take a set of $n+2$ elements. The richness property implies that under two distinct distance functions d_1, d_2 the clustering function f may form two clusterings, respectively Γ_1, Γ_2, where the first n elements form one cluster and in Γ_1 the remaining two elements are in one cluster, and in Γ_2 they are in two separate clusters. By the invariance property, we can derive from d_2 the distance function d_4 such that no distance between the elements under d_4 is lower than the biggest distance under d_1. By the invariance property, we can derive from d_1 the distance function d_3 such that the distance between elements $n+1, n+2$ is bigger than under d_4. We have then $f(\{1,..,n+2\}; d_4) = \Gamma_2, f(\{1,..,n+2\}; d_3) = \Gamma_1$. Now let us apply the consistency axiom. From d_4 we derive the distance function d_6 such that for elements $1, ..., n$ the d_1 and d_6 are identical, the distance between $n+1, n+2$ is the same as in d_4, and the distances between any element of $1, ..., n$ and any of $n+1, n+2$ is some l that is bigger than any distances between any elements under $d_1, ..., d_4$. From d_3 we derive the distance function d_5 such that for elements $1, ..., n$ the d_1 and d_5 are identical, the distance between $n+1, n+2$ is the same as in d_4 and the distances between any element of $1, ..., n$ and any of $n+1, n+2$ is the same l as above. We have, then, $f(\{1,..,n+2\}; d_6) = \Gamma_2, f(\{1,..,n+2\}; d_5) = \Gamma_1$. But this means a contradiction, because by construction, d_5 and d_6 are identical.

At the same time, Kleinberg demonstrated that there exist algorithms that satisfy any pair of the above conditions. He uses for the purpose of this demonstration the versions of the well known statistical single-linkage procedure. The versions differ by the stopping condition:

- k-cluster stopping condition (which stops adding edges as soon as the linkage graph for the first time consists of k connected components) - not "rich"

- distance-r stopping condition (which adds edges of weight at most r only) - not scale-invariant
- scale-stopping condition (which adds edges of weight being at most some percentage of the largest distance between nodes) - not consistent

Notice that also k-median and k-means, as well as the MDL clustering do not fulfill the consistency axiom.

F.1.2 Common sense problems

As there are practical limitations on the measurement precision, scale-invariance may not be appropriate for the whole range of scaling factors.

The consistency axiom precludes a quite natural phenomenon that under stretching of distances new clusters may be revealed.

Finally, we are really not interested in getting clusters of cardinality say equal one, so that the richness axiom is in fact not intuitive,

F.1.3 Clusterability theory concerns

A set of objects is not all we want to know when saying that we discovered a cluster. We want to see that objects belonging to different clusters differ substantially from one another, for example that the clusters are separated from one another by some space.

But this separating space is something what the majority of clustering methods do not take into account and about which Kleinberg's axiomatisation does not care. Speaking differently, a clustering algorithm may provide us with clusters even when there are none in the data.

The very existence of clusters in the data, or, more precisely, the separation of clusters, was the subject of research on the so-called clusterability[2], see e.g. [3, 2].

Definition F.1.2 *[3] Clusterability is a function of the set $X \subset \mathbb{R}^m$ mapping it into the set of real numbers that specifies how a set X is clusterable.*

These functions are constructed in such a way that the higher the value of the function the stronger is the evidence that high quality clusterings may occur.

Methods of cluster analysis, examined by Ackerman [3], on which we base our reflection in this section, are limited to the so-called center based (or centric) clustering,

Definition F.1.3 *We say that a partition is a* centric clustering *if each cluster is distinguished by the center or several centers, where the distance of any element to the nearest center of its own cluster is not greater than that to any other center (of any other cluster).*

[2] Instead of clusterability, that is, the issue of the very existence of clusters, the cluster validity, that is - their agreement with some prior knowledge about expected clusters, may be investigated, as discussed already in chapter 4. Cluster validity was also studied in papers [81, 264].

Centric clusterings are a special case of results from the distance driven clustering functions (Definition F.1.1) so that Kleinberg's axioms should be applicable.

To express the inadequacy of data clustering, Ackerman and Ben-David introduce the concept of *loss function* for this class of clustering. An optimal clustering minimizes the loss function for a particular data set.

Ackerman introduced the following concepts:

Definition F.1.4 *Two centric clusterings are ϵ-close, if you can create a set of disjoint pairs of centers from both clusterings such that the distance between the centers of each pair is less than ϵ.*

Definition F.1.5 *Data is ϵ, δ-clusterable if the loss function for any clustering, that is ϵ-close to an optimal clustering, is not greater than $(1+\delta)$ times the value of the loss function of the optimal clustering (perturbational clusterability).*

The study [3] proposed algorithms detecting such clusterability for $\epsilon = \frac{radius(X)}{\sqrt{l}}$, where l is the cardinality of subsets of X, which all have to be considered.

Let us recall two concepts on which clusterability evaluation is based.

Definition F.1.6 *The* separation in the clustering *is the minimum distance between the elements of different classes.* Clustering diameter *is the maximum distance between elements of the the same class.*

It is argued in [3] that there is at most one clustering such that the separation is larger than the diameter.

With these concepts, Ackerman introduces in [3] various kinds of clusterability.

Definition F.1.7 *(Clusterability and related concepts)*

- Worst-pair-quality of a clustering *is the ratio of separation and diameter.*
- Worst-pair-clusterability *is the minimal worst-pair-quality over all centric clusterings of the set X.*
- k-separable clusterability *is the decrease in the loss function while passing from $k-1$ clusters to k clusters.*
- Variance-clusterability *is the quotient of the variance between clusters to the variance within clusters.*
- Target-clusterability *is measured as the distance to the clustering defined manually.*

It is claimed that in polynomial time, one can calculate only the worst-pair-clusterability.

Note, first of all, that all these clusterability measures are sensitive to outliers because (most) clustering methods do not allow for an element to be left unclustered.

But more interesting is the relationship to the Kleinberg's axioms. We would naively expect that if a clustering algorithm returns a clustering, then there exists an intrinsic clustering. Furthermore, one would expect that if there is an intrinsic clustering, then the clustering algorithm returns it. And it returns (approximately) the clustering that is there in the data.

But any algorithm, seeking clusters fitting the requirements of ϵ, δ-clusterability, will fail under distance-scaling, so it will not satisfy the Kleinberg's invariance-axiom.

On the other hand, a clustering algorithm, seeking to match the worst-pair-clusterability or variance-clusterability or target-clusterability criterion, will perform well under scaling.

The k-separable clusterability may or may not depend on scaling this being determined by the loss function.

The axiom of consistency seems not to constitute any obstacle in achieving any of the mentioned clusterability criteria. However, the Kleinberg's Γ-transformation may lead to emergence of new clusters according to at least the variance-based clusterability criterion.

The Kleinberg's axiom of richness is, however, hard to fulfil at least by the variance-based clusterability. This clusterability criterion puts preference on clusters with small numbers of elements, but if the variance is to be estimated with any basic scrutiny, the minimum number of three elements within each cluster is necessary, which precludes matching of the richness axiom.

F.1.4 Learnability theory concerns

Learnability theory [342] defines learnability as the possibility to generalize from the sample to the population. Its basic postulate is: the concept is learnable if it can be falsified. In order to learn, the algorithm must be able to conclude that the data do not belong to the space of concepts. If the algorithm is able to assign to each set of data a concept from the concept space, it does not learn anything.

In the light of the learnability theory

- the classical cluster analysis does not reveal any "natural clusters", but provides a mixture of the intrinsic structure of the data and of the structure induced by the clustering algorithm
- the classical cluster analysis is not a method of learning without supervision ("without a teacher")[3]
- the capability of a clustering algorithm to recreate the external ("manual") clustering is not a criterion for the correctness of the algorithm, if it has not learning abilities
- if the cluster analysis has something to contribute, it cannot satisfy the axioms of Kleinberg that is to
  - produce any clustering possible (via manipulation of distance)
  - produce invariant results while scaling distance
  - produce invariant results with respect to reduction of distance to cluster center.

[3] There is, in fact, always a teacher who provides an aesthetic criterion of what is "similar"/"different"

because these axioms give a much too big number of degrees of freedom for manipulation of the function of distance.

F.2 Cluster quality axiomatisation

Ackerman and Ben-David[4] propose to resolve the problem of Kleinberg's axiomatics by axiomatising not the clustering function, but rather cluster quality function. We base the following on their considerations.

Definition F.2.1 *Let $\mathfrak{C}(\mathbf{X})$ be the set of all possible clusterings over the set of objects $\mathbf{X}$, and let $\mathfrak{D}(\mathbf{X})$ be the set of all possible distance functions over the set of objects $\mathbf{X}$.*

A clustering-quality measure (CQM) $J : \mathbf{X} \times \mathfrak{C}(\mathbf{X}) \times \mathfrak{D}(\mathbf{X}) \rightarrow \mathbb{R}^+ \cup \{0\}$ is a function that, given a data set (with a distance function) and its partition into clusters, returns a non-negative real number representing how strong or conclusive the clustering is.

Ackerman and Ben-David propose the following axioms:

Axiom F.2.1 *(Scale Invariance) A quality measure J satisfies scale invariance if for every clustering Γ of (X, d)[5], and every positive β, $J(X, \Gamma, d) = J(X, \Gamma, \beta d)$.*

Note that if we define a clustering function in such a way that it maximises the quality function, then the clustering function has also to be invariant.

Axiom F.2.2 *(Consistency) A quality measure J satisfies richness if for every clustering Γ over (X, d), whenever d' is a Γ-transformation of d, then $J(X, \Gamma, d') \geq J(X, \Gamma, d)$.*

Note that if we define a clustering function in such a way that it maximises the quality function, then the clustering function *does not need to be invariant.*

Axiom F.2.3 *(Richness) A quality measure m satisfies richness if for each non-trivial clustering Γ^* of X, there exists a distance function d over X such that $\Gamma^* = argmax_\Gamma\{J(X, \Gamma, d)\}$.*

Note that if we define a clustering function in such a way that it maximises the quality function, then the clustering function has also to be rich.

Ackerman and Ben-David claim that

Theorem F.2.1 *Consistency, scale invariance, and richness for clustering-quality measures form a consistent set of requirements.*

[4] M. Ackerman and S. Ben-David: Measures of Clustering Quality: A Working Set of Axioms for Clustering. in: D. Koller and D. Schuurmans and Y. Bengio and L. Bottou eds: Advances in Neural Information Processing Systems 21 NIPS09, 2009, pp. 121–128

[5] X is the set of objects, d is a distance function

and prove this claim by providing a quality measure that matches all these axioms.

The quality measure they propose here is the ”Relative Margin”. First, one needs to compute the ratio of distance of a data point to its closest centre to that to the second closest centre. Then, a representative set of a clustering is defined as a set K containing exactly one element from each cluster. One computes the average of the ratio, mentioned before, for each potential representative set. The minimum over these averages is the cluster quality function called Relative Margin. The lower the value of the Relative Margin, the higher the clustering quality. This does not match quite their axiomatisation because the axiomatisation assumed that increasing quality is related to increasing quality function value. So to satisfy the axiomatic system, axioms have to be inverted, so that quality increase is related to function decrease.

A disadvantage of this measure is that it ranks highly a clustering in which each element is a separate cluster.

F.3 Relaxations for overcoming the Kleinberg's problems

In the preceding section one way of clustering axiomatisation improvement was presented. It consisted in shifting axiomatisation from clustering function to clustering quality, which imlicitly relaxed at least one Kleinberg's axiom (consistency).

Let us look now at other proposals, concerning now the explicit relaxation of the Kleinberg's axioms, as summarised by Ben-David.

Axiom F.3.1 *(Relaxation of Kleinberg's richness) For any partition $\mathit{\Gamma}$ of the set $\mathbf{X}$, consisting of exactly k clusters, there exists such a distance function d that the clustering function $f(d)$ returns this partition $\mathit{\Gamma}$.*

This relaxation allows for the algorithms splitting the data into a fixed number of clusters, like k-means. But it still leaves the open problem of a minimal number of elements in a cluster.

Axiom F.3.2 *(Local Consistency) Let $C_1, \dots, C_k$ be the clusters of $f(d)$. For every $\beta_0 \geq 1$ and positive $\beta_1, \dots \beta_k \leq 1$, if d' is defined by: $d'(a,b) = \beta_i d(a,b)$ for a and b in C_i, $d'(a,b) = \beta_0 d(a,b)$ for a, b not in the same $f(d)$-cluster, then $f(d) = f(d')$.*

This axiom does not guarantee that d' is in fact a distance, so that it is hard to satisfy.

Axiom F.3.3 *(Refinement Consistency) is a modification of the consistency axiom obtained by replacing the requirement that $f(d) = f(d')$ with the requirement that one of $f(d)$, $f(d')$ is a refinement of the other.*

Obviously, the replacement of the consistency requirement with refinement consistency breaks the impossibility proof of Kleinberg's axiom system. But there is a practical concern: assume a data set of points uniformly randomly distributed in a plane on line segments $((a,0),(2a,6a))$, $((a,0),(2a,-6a))$, $((-a,0),(-2a,6a))$, $((-a,0),(-2a,-6a))$. A k-means algorithm with $k=2$ would create two clusters: (1) segments $((a,0),(2a,6a))$, $((a,0),(2a,-6a))$, (2) $((-a,0),(-2a,6a))$, $((-a,0),(-2a,-6a))$. But if cluster (2) is shrunk as follows: all points are rotated around $(-a,0)$ so that they lie on the X axis to the left of $(-a,0)$, then k-means would change class allocation of a part of points from this cluster, violating refinement consistency.

F.4 Learnability oriented axiomatisation

Let us now look at ways to formulate a learnability based axiomatic framework for clustering algorithms. The clustering algorithm is inevitably connected to the clusterings it can produce. So, first let us state the axioms necessary for clustering.[6]

Axiom F.4.1 *(Strong learnability) Clustering must split the sample space (for each point in sample space we must be able to say to which cluster it belongs or if it belongs to the undecided space),*

Axiom F.4.2 *(Weak learnability) Clustering must be learnable from finite sample (the clustering to be discovered must belong to a class of clusterings such that the class is learnable in the sense of learnability theory),*

Axiom F.4.3 *(Separability) Clusters in the clustering must be separable (for a sufficiently large sample it should be significantly more probable to have the closest neighbor from the same cluster than from a different one),*

Axiom F.4.4 *(Enlightening) Clustering must be enlightening (new information should be obtained via clustering compared to prior knowledge).*

Therefore:

Axiom F.4.5 *(Learnability) A clustering algorithm with high probability shall return a clustering if the intrinsic clustering of the data belongs to the class of clusterings for which the algorithm is designed and this clustering should be close to the intrinsic one,*

Axiom F.4.6 *(Strong learnability) A clustering algorithm with high probability shall return a failure information if the intrinsic clustering of the data does not belong to the class of clusterings for which the algorithm is designed or if there is no clustering behind the data,*

Axiom F.4.7 *(Separability) A clustering algorithm shall state with high reliability what is the separation between clusters,*

[6] See: R. A. Kłopotek and M. A. Kłopotek: Fallacy of Kleinberg's Richness Axiom in Document Clustering. *Conference on Tools, Applications and Implementations of Methods For Determining Similarities Between Documents.* Warszawa, 22.4.2015

Axiom F.4.8 *(Enlightening) A clustering algorithm shall verify if there is a difference between the detected clustering and the prior knowledge of the sample space.*

Note that these axioms are in opposition to axioms of Kleinberg. Learnability contradicts the richness axiom. Separability contradicts invariance. Enlightening opposes consistency.

F.5 Graph clustering axiomatisation

In the preceding sections we considered the case, when the clustered objects are placed in a space, where distances between objects may be defined. But a large portion of this book was devoted to clustering of graphs (chapter 5).

Hence we cannot overlook axiomatic frameworks developed specially for them.

Van Laarhoven and Marchiori[7] developed an axiomatic framework, defining the desired properties of the clustering quality function, understood in a similar way as previously.

Their first axiom assumes "Permutation invariance", namely.

Axiom F.5.1 *(Permutation invariance) The graph partition quality depends on node similarities but not on node labels.*

The second axiom is the "Scale invariance" which means that

Axiom F.5.2 *(Scale invariance) Proportional increase/decrease of weights of edges (node similarities) does not change the partition quality.*

Authors quoted request also the "Richness", that is

Axiom F.5.3 *(Richness) For any partition of the sets of nodes there exists a graph, for which this partition is optimal with respect to the given quality function among all the partitions of the graph considered.*

Further, they request the "Consistent improvement monotonicity", that is

Axiom F.5.4 *(Consistent improvement monotonicity) If the edge weights within clusters are increased and between clusters they are decreased, then the quality function shall increase.*

One sees immediately that these axioms parallel those from section F.2 – the difference is that we talk now about similarity measures and not distances and that some similarities could be equal zero. A similarity equal to zero breaks Kleinberg's contradiction proof, because scaling by the lowest and highest distances is necessary, and in this case the highest distance is infinite. However, there are two difficulties here. One is that the graph may have all edges with

[7] see: T. van Laarhoven and E. Marchiori: Axioms for Graph Clustering Quality Functions, Journal of Machine Learning Research 15 (2014) 193-215

non-zero weights. In this case the claim of breaking Kleinberg's contradiction is not valid. Furthermore, the weight zero stems frequently from the fact of thresholding the similarity level between nodes. In this case the Scale invariance is violated, because an increase in similarity would introduce new edges, its decrease would remove the existing ones (crossing the threshold).

Van Laarhoven and Marchiori introduce, as well, an axiom for "Local consistency" (violated by the algorithms with a fixed number of clusters):

Axiom F.5.5 *(Local consistency) For graphs agreeing on some subgraphs, if a change of clustering in the common subgraph increases the quality function in one graph, then the same change should cause an increase in the other.*

Their final axiom is called "Continuity". That is

Axiom F.5.6 *(Continuity) For each $\epsilon > 0$ there exists a $\delta > 0$ such that for two graphs differing by edge weights for each edge by only δ, the quality functions of the two graphs differ only by ϵ for each clustering C (of both graphs).*

The same authors show that the popular graph partition quality measure called modularity[8] conforms only to four of their axioms (richness, continuity, scle invariance, permutation invariance) but violates monotonicity and locality.

Therefore, they propose a new quality function, called adaptive scale modularity,

$$Q_{M,\gamma} = \sum_{c \in \Gamma} \left(\frac{w(c)}{M + \gamma vol(c)} - \left(\frac{vol(c)}{M + \gamma vol(c)} \right)^2 \right)$$

where $vol(c)$ is the volume of the cluster c, and $w(c)$ is the sum of weights of edges within the cluster c. This function satisfies all their axioms for $M = 0, \gamma \geq 2$, and violates only scale invariance for $M > 0$.

For comparison, the original modularity was of the form

$$Q_{M,\gamma} = \sum_{c \in \Gamma} \left(\frac{w(c)}{vol(G)} - \left(\frac{vol(c)}{vol(G)} \right)^2 \right)$$

where $vol(G)$ is the volume of the entire graph G.

It has been observed by Fortunato and Barthelemy[9] that modularity suffers from the so-called resolution limit. Consider a ring of cliques in which cliques are interconnected by a single link only. One would expect that an optimal clustering would contain exactly one clique in one cluster. However, optimality with respect to modularity does not behave in this way. Therefore, Fortunato and Barthelemy require that graph partition quality measure should be Resolution-limit-free. This means that: If a partition Γ of a graph G is optimal with respect to such a measure among all partitions of G, then for each subgraph G' of G the

[8] See: Mark E. J. Newman and Michelle Girvan. Finding and evaluating community structure in networks. Phys. Rev. E, 69:026113, Feb 2004.

[9] See: Santo Fortunato and Marc Barthelemy. Resolution limit in community detection. Proc. Natl. Acad. Sci. USA, 104(1):36-41, 2007.

partition Γ' induced from Γ by G' should also be optimal among all partitions of G'.

Adaptive scale modularity does not fulfil this requirement but it does not suffer, nonetheless, from the resolution limit. Hence, the axiom of resolution-limit-freedom remains an open question for further investigations.

Let us make a further remark on modularity. It has a clear interpretation as a difference between the current link structure and a random one. This is in concordance with the enlightening axiom, mentioned before. The adaptive scale modularity lacks such an interpretation and one can suspect that it will promote big clusters (high volume compared to border, that is the sum of cluster node degrees being much larger than the number of edges linking the given cluster with other clusters). Hence, it seems that lack of contradictions in an axiomatic system is a necessary condition for the adequacy of this system, but it is not sufficient.

Furthermore, it turns out that finding a partition optimising the value of modularity is NP hard. Hence, in fact approximate, but fast algorithms are used instead, without even any guarantees of giving a solution within some bounds with respect to the optimal one.

Therefore, it seems reasonable to press on learnability rather than on richness in axiomatic systems.

This concluding remark would be conform with the opinion of Estivill-Castro, expressed in his position paper [10] "[...] there are many clustering algorithms, because the notion of "cluster" cannot be precisely defined. Clustering is in the eye of the beholder". That is we have to define our notion of clustering and then prove that it is learnable from data using one's algorithm.

[10] Vladimir Estivill-Castro: Why So Many Clustering Algorithms: A Position Paper. SIGKDD Explor. Newsl., June 2002, Vol. 4, No. 1, 65–75.

References

1. Z. Abbassi and V.S. Mirrokni. A recommender system based on local random walks and spectral methods. In *Proc. of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis*, pages 102–108, San Jose, California, USA, 12-17 Aug. 2007. ACM New York, NY, USA. [cited on page 53]
2. M. Ackerman and S. Ben-David. Which data sets are 'clusterable'? – a theoretical study of clusterability. preprint (D.R.C. School of Computer Science, University of Waterloo), 2008. URL: `http://www.ima.umn.edu/~iwen/REU/ability\_submit.pdf`. [cited on page 67, 269]
3. M. Ackerman and S. Ben-David. Clusterability: A theoretical study. In *Proc. of the 12th International Conference on Artificial Intelligence and Statistics (AISTATS 2009), JMLR: W&CP 5*, pages 1–8, 2009. [cited on page 269, 270]
4. C.C. Aggarwal, A. Hinneburg, and D.A. Keim. On the surprising behavior of distance metrics in high dimensional space. In *Proc. of the 8th Int. Conf. on Database Theory*, LNCS 1973, pages 420–430. Springer-Verlag Berlin, Heidelberg, 2001. [cited on page 27, 28, 242]
5. Ch.C. Aggarwal and Ch.K. Reddy, editors. *Data Clustering. Algorithms and Applications.* CRC Data Mining and Knowledge Discovery Series. Chapman and Hall/CRC, Boca Raton, FL, 2013. [cited on page 21]
6. M.A. Ajzerman, E.M. Brawerman, and L.I. Rozonoer. *Rozpoznawanie obrazów. Metoda funkcji potencjałowych.* WNT, Warszawa, 1976. [cited on page 18, 57]
7. N. Alldrin, A. Smith, and D. Turnbull. Clustering with EM and k-means. Technical report, Department of Computer Science. University of California, San Diego. La Jolla, CA 92037, 2003. URL: `http://neilalldrin.com/research/w03/cse253/project1.pdf`. [cited on page 97]
8. D. Aloise, A. Deshpande, P. Hansen, and P. Popat. NP-hardness of Euclidean sum-of-squares clustering. *Cahiers du GERAD*, pages G–2008–33, 2008. [cited on page 42]
9. N. Alon. Eigenvectors and expanders. *Combinatorica*, 6(2):83–96, 1986. DOI: 10.1007/BF02579166. [cited on page 154]
10. M.R. Anderberg. *Cluster Analysis for Applications.* Academic Press, London, 1973. [cited on page 18, 21, 39]
11. R. Andersen, Ch. Borgs, J. Chayes, J. Hopcraft, V.S. Mirrokni, and S.-H. Teng. Local computation of PageRank contributions. *Internet Mathematics*, 5(1-2):23–45, 2008. DOI: 10.1080/15427951.2008.10129302. [cited on page 211]
12. R. Andersen and F. Chung. Detecting sharp drops in PageRank and a simplified local partitioning algorithm. In J. Cai, S.B. Cooper, and H. Zhu, editors, *Proc. 4th Int. Conf. on Theory and Applications of Models of Computation, TAMC 2007*, volume 4484 of *LNCS*, pages 1–13, Shanghai, China, 22-25 May 2007. Springer. DOI: 10.1007/978-3-540-72504-6_1. [cited on page 266]
13. R. Andersen, F. Chung, and K. Lang. Using PageRank to locally partition a graph. *Internet Mathematics*, 4(1):35–64, 2007. [cited on page 205, 209, 262, 264, 265]

14. R. Andersen, F. Chung, and K. Lang. Local partitioning for directed graphs using PageRank. *Internet Mathematics*, 5(1-2):3–22, 2008. [cited on page 19, 205, 211]
15. R. Andersen and S.M. Cioabă. Spectral densest subgraph and independence number of a graph. *J. of Universal Computer Science*, 13(11):1501–1513, Nov. 2007. DOI: 10.3217/jucs-013-11-1501. [cited on page 211]
16. R. Andersen and K.J. Lang. Communities from seed sets. In *Proc. of the 15th Int. Conf. on World Wide Web*, WWW'06, pages 223–232, Edinburgh, Scotland, 23-26 May 2006. ACM Press, New York, NY. DOI: 10.1145/1135777.1135814. [cited on page 205]
17. M. Ankerst, M.M. Breunig, H.-P. Kriegel, and J. Sander. OPTICS: Ordering points to identify the clustering structure. In *Proc. of 1999 ACM-SIGMOD Int. Conf. on Management of Data*, pages 49–60, Philadelphia, Pennsylvania, Jun. 1999. ACM Press. [cited on page 54]
18. A. Anthony and M. des Jardins. Open problems in relational data clustering. In *Proc. ICML Workshop on Open Problems in Statistical Relational Learning*, Pittsburgh, PA, 2006. [cited on page 52]
19. P. Arbenz. Lecture notes on solving large scale eigenvalue problems. D-Infk., ETH Zürich, 2012. URL: `http://people.inf.ethz.ch/arbenz/ewp/lnotes.html`. [cited on page 154]
20. D. Arthur. *Analyzing and improving local search: k-means and ICP*. PhD thesis, Stanford University. Department of Computer Science, Stanford, CA, Jun. 2009. [cited on page 42, 71, 72, 77]
21. D. Arthur, B. Manthey, and H. Röglin. Smoothed analysis of the k-means method. *J. of the ACM (JACM)*, 58(5):19, Oct. 2011. DOI: 10.1145/2027216.2027217. [cited on page 70]
22. D. Arthur and S. Vassilvitskii. k-means++: the advantages of careful seeding. In N. Bansal, K. Pruhs, and C. Stein, editors, *Proc. of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA 2007, pages 1027–1035, New Orleans, Louisiana, USA, 7-9 Jan. 2007. SIAM. [cited on page 13, 70, 77, 78]
23. S. Arya, D.M. Mount, N.S. Netanyahu R. Silverman, and A.Y. Wu. An optimal algorithm for approximate nearest neighbor searching. *J. ACM*, 45(6):891–8923, Nov. 1998. DOI: 10.1145/293347.293348. [cited on page 79]
24. B. Aspvall and J.R. Gilbert. Graph coloring using eigenvalue decomposition. *SIAM. J. on Algebraic and Discrete Methods*, 5(4):526–538, Dec. 1984. [cited on page 160]
25. A. Asuncion and D.J. Newman. UCI machine learning repository. University of California, Irvine, School of Information and Computer Sciences, 2007. URL: `http://www.ics.uci.edu/~mlearn/MLRepository.html`. [cited on page 102]
26. K. Avrachenkov, V. Dobrynin, D. Nemirovsky, S.K. Pham, and E. Sirnova. PageRank based clustering of hypertext document collections. In *Proc. of the 31st Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, Singapore, Singapore*, SIGIR '08, pages 873–874. ACM, New York, NY, USA, 20-24 Jul. 2008. DOI: 10.1145/1390334.1390549. [cited on page 220, 222]
27. R. Babuška. *Fuzzy Modeling for Control*. Kluwer Academic Publishers, Boston, USA, 1998. [cited on page 112]
28. R. Babuška, P.J. van der Veen, and U. Kaymak. Improved covariance estimation for Gustafson-Kessel clustering. In *Proc. of the 2002 IEEE Int. Conf. on Fuzzy Systems, 2002. FUZZ-IEEE'02*, volume 2, pages 1081–1085. IEEE, 12-17 May 2002. DOI: 10.1109/FUZZ.2002.1006654. [cited on page 112]
29. R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press/Addison-Wesley, New York, 1999. [cited on page 31, 217]

30. B. Bahmani, B. Moseley, A. Vattani, R. Kumar, and S. Vassilvitskii. Scalable k-means++. *Proceedings of the VLDB Endowment*, 5(7):622–633, Mar. 2012. [cited on page 78]
31. Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst. *Templates for the Solution of Algebraic Eigenvalue Problems: a Practical Guide*. SIAM, Philadelphia, Jan. 2000. [cited on page 238]
32. M. F. Balcan, A. Blum, and N. Srebro. A theory of learning with similarity functions. *Machine Learning*, 72(1-2):89–112, Aug. 2008. DOI: 10.1007/s10994-008-5059-5. [cited on page 23]
33. M. F. Balcan, A. Blum, and S. Vempala. Clustering via similarity functions: Theoretical foundations and algorithms. Unpublished, 2009. URL: `http://www.cs.cmu.edu/afs/cs.cmu.edu/Web/People/avrim/Papers/BBVclustering_full.pdf`. [cited on page 23]
34. P. Baldi and G.W. Hatfield. *DNA Microarrays and Gene Expression: From Experiments to Data Analysis and Modeling*. Cambridge University Press, Cambridge, UK, 2002. [cited on page 17]
35. G.H. Ball and D.J. Hall. ISODATA: a novel method of data analysis and pattern classification. Technical Report NTIS AD 699616, Stanford Research Institute, Stanford, CA, 1965. [cited on page 69, 80]
36. A. Banerjee, C. Krumpelman, J. Ghosh, S. Basu, and R.J. Mooney. Model-based overlapping clustering. In *Proc. of the 11th ACM SIGKDD Intl Conf. on Knowledge Discovery in Data Mining*, pages 532–537. ACM New York, NY, USA, Aug. 2005. DOI: 10.1145/1081870.1081932. [cited on page 50]
37. A. Banerjee, S. Merugu, I.S. Dhillon, and J. Ghosh. Clustering with Bregman divergences. *J. Mach. Learn. Res.*, 6:1705–1749, 1 Dec. 2005. [cited on page 30, 31, 44]
38. W. Barbakh and C. Fyfe. Online clustering algorithms. *Int. J. Neural Syst.*, 18(3):185–194, 2008. DOI: 10.1142/S0129065708001518. [cited on page 81]
39. S.T. Barnard, A. Pothen, and H.D. Simon. A spectral algorithm for envelope reduction of sparse matrices. *Numerical Linear Algebra with Applications*, 2(4):317–334, Jul./Aug. 1995. DOI: 10.1002/nla.1680020402. [cited on page 145]
40. S.T. Barnard and H.D. Simon. Fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems. *Concurrency – Practice and Experience*, 6(2):101–117, Apr. 1994. DOI: 10.1002/cpe.4330060203. [cited on page 156]
41. G. Barnett. Correspondence analysis: A method for the description of communication networks. In W.D. Richards and G. Barnett, editors, *Progress in Communication Sciences*, volume 12, pages 135–164. Ablex Publishing Corporation, U.S., 1993. [cited on page 169]
42. M. Basseville. Distance measures for signal processing and pattern recognition. *Signal Processing*, 18(4):349–369, Dec. 1989. doi: 10.1016/0165-1684(89)90079-0. [cited on page 30]
43. M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 6(15):1373–1396, Jun. 2003. [cited on page 147, 148]
44. A. Ben-Hur, D. Horn, H.T. Siegelmann, and V. Vapnik. Support vector clustering. *J. of Machine Learning Research 2 (2001) 125-137*, 2:125–137, 2001. [cited on page 60]
45. A.M. Bensaid, L.O. Hall, J.C. Bezdek, L.P. Clarke, M.L. Silbiger, J.A. Arrington, and R.F. Murtagh. Validity-guided (re)clustering with applications to image segmentation. *IEEE Trans. Fuzzy Syst.*, 4(2):112–123, 1996. [cited on page 102]

46. P. Berkhin. Bookmark-coloring approach to personalized PageRank computing. *Internet Mathematics*, 3(1):41–62, 2006. DOI: 10.1080/15427951.2006.10129116. [cited on page 264]
47. P. Berkhin. A survey of clustering data mining techniques. In J. Kogan, Ch. Nicholas, and M. Teboulle, editors, *Grouping Multidimensional Data*, pages 25–72. Springer, 2006. [cited on page 21]
48. M.W. Berry, M. Browne, A.M. Langville, V.P.Pauca, and R.J. Plemmons. Algorithms and applications for approximate nonnegative matrix factorization. *Computational Statistics & Data Analysis*, 52(1):155–173, 2007. [cited on page 46]
49. M.W. Berry, S.T. Dumais, and G.W. O'Brien. Using linear algebra for intelligent information retrieval. *SIAM Review*, 37(4):573–595, Dec. 1995. DOI: 10.1137/1037127. [cited on page 53]
50. J.C. Bezdek. A convergence theorem for the fuzzy ISODATA clustering algorithms. *IEEE Trans. Pattern Anal. Mach. Intell*, PAMI-2(1):1–8, 1980. [cited on page 100, 104]
51. J.C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms.* Plenum Press, New York and London, 1981. [cited on page 18, 21, 100, 105, 106, 109]
52. J.C. Bezdek and R.J. Hathaway. Numerical convergence and interpretation of the fuzzy c-shells clustering algorithm. *IEEE Trans. on Neural Networks*, 3(5):787–793, Sep. 1992. DOI: 10.1109/72.159067. [cited on page 115]
53. J.C. Bezdek, J. Keller, R. Krisnapuram, and N.R. Pal. *Fuzzy Models and Algorithms for Pattern Recognition and Image Processing.* Kluwer Academic Publisher, Boston, London, 1999. [cited on page 113, 124]
54. J.C. Bezdek and S.K. Pal. *Fuzzy Models for Pattern Recognition: Metods that Search for Structures in Data.* IEEE, New York, 1992. [cited on page 21, 99, 109]
55. C.M. Bishop. *Neural Networks for Pattern Recognition.* Clarendon Press, London, 1995. [cited on page 21, 95]
56. Ch. Boutsidis, A. Zouzias, M.W. Mahoney, and P. Drineas. Stochastic dimensionality reduction for k-means clustering. `arXiv:1110.2897v1 [cs.DS]`, 13 Oct 2011. [cited on page 128]
57. P.S. Bradley, K.P. Bennett, and A. Demiriz. Constrained k-means clustering. Technical Report MSR-TR-2000-65, Microsoft Research, May 2000. [cited on page 80, 91]
58. P.S. Bradley and O.L. Mangasarian. k-plane clustering. *J. of Global Optimization*, 16(1):23–32, 2000. [cited on page 114]
59. M. Brand. A random walk perspective on maximizing satisfaction and profit. Technical Report TR2005-050, Mitsubishi Electric Research Laboratories, Dec. 2005. URL: `http://www.merl.com/reports/docs/TR2005-050.pdf`. [cited on page 191, 198]
60. J.S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proc. 14-th Conf. on Uncertainty in Artificial Intelligence*, pages 43–52, Madison WI, 1998. Morgan Kauffman. [cited on page 33]
61. M. Breitenbach and Gr.Z. Grudic. Clustering through ranking on manifolds. In *Proc. of the 22nd Intnl Conf. on Machine Learning, ICML'05*, pages 73–80. ACM New York, NY, USA, 2005. DOI: 10.1145/1102351.1102361. [cited on page 214]
62. A.E. Brouwer and W.H. Haemers. *Spectra of Graphs.* Springer, 2011. [cited on page 247]
63. J.-Ph. Brunet, P. Tamayo, T.R. Golub, and J.P. Mesirov. Metagenes and molecular pattern discovery using matrix factorization. *PNAS*, 101(12):4164–4169, Mar. 2004. DOI: 10.1073/pnas.0308531101. [cited on page 63, 64]

64. S. Bubeck, M. Meilă, and U. von Luxburg. How the initialization affects the stability of the k-means algorithm. `arXiv:0907.5494v1 [stat.ML]`, 31 Jul. 2009. [cited on page 77]
65. Th. Bühler and M. Hein. Spectral clustering based on the graph p-laplacian. In *Proc. of the 26th Annual Intl Conf. on Machine Learning, ICML'09*. ACM, New York, NY, USA, 2009. doi: 10.1145/1553374.1553385. [cited on page 161]
66. S.H. Cha. Comprehensive survey on distance/similarity measures between probability density functions. *Intl J. of Mathematical Models and Methods in Applied Sciences*, 4(1):300–307, 2007. [cited on page 26]
67. P.K. Chan, M.D.F. Schlag, and J.Y. Zien. Spectral K-way ratio-cut partitioning and clustering. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 13(9):1088–1096, Sep. 1994. [cited on page 160, 173]
68. A.K. Chandra, P. Raghavan, W.L. Ruzzo, R. Smolensky, and P. Tiwari. The electrical resistance of a graph captures its commute time and cover times. *Comput. Complexity*, 6(4):312–340, 1997. [cited on page 197, 257]
69. M. Charikar, C. Chekuri, T. Feder, and R. Motwani. Incremental clustering and dynamic information retrieval. In *Proc. of 29th Symposium on Theory of Computing*, pages 626–635. ACM, New York, 1997. [cited on page 132]
70. P. Cheeseman and J. Stutz. Bayesian classification (autoclass): Theory and results. In U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 153–180. AAAI Press/MIT Press, 1996. [cited on page 96]
71. H. Chen and J. Peng. 0-1 semidefinite programming for graph-cut clustering: Modelling and approximation. In P.M. Pardalos and P. Hansen, editors, *Data mining and mathematical programming*, CRM Proceedings & Lecture Notes, pages 15–41. AMS, 2008. [cited on page 45]
72. S. Chen, B. Ma, and K. Zhang. On the similarity metric and the distance metric. *Theoretical Computer Science*, 410(24-25):2365–2376, May 2009. doi: 10.1016/j.tcs.2009.02.023. [cited on page 26]
73. T.W. Cheng, D.B. Goldof, and L.O. Hall. Fast fuzzy clustering. *Fuzzy Sets and Systems*, 93(1):49–56, Jan. 1998. [cited on page 109]
74. H. Choe and J. Jordan. On the optimal choice of parameters in a fuzzy c-means algorithm. In *Proc. First IEEE Conf. on Fuzzy Systems*, pages 349–353, San Diego, CA, 1992. [cited on page 102]
75. F. Chung. *Spectral Graph Theory*. AMS, Providence, RI, 1997. [cited on page 19, 145, 147, 162, 200, 247, 249]
76. F. Chung. Laplacians and the Cheeger inequality for directed graphs. *Annals of Combinatorics*, 9(1):1–19, Apr. 2005. DOI: 10.1007/s00026-005-0237-z. [cited on page 250, 251]
77. F. Chung. Random walks and local cuts in graphs. *Linear Algebra and its Applications*, 423(1):22–32, 1 May 2007. [cited on page 19, 251, 255]
78. F. Chung. Graph theory in the information age. *Notices of the AMS*, 57(6):726–732, 2010. [cited on page 145, 200]
79. F. Chung and A. Tsiatas. Finding and visualizing graph clusters using PageRank optimization. In R. Kumar and D. Sivakumar, editors, *Algorithms and Models for the Web Graph. 7th Int. Workshop, WAW 2010. Stanford, CA, USA*, volume 6516 of *LNCS*, pages 86–97. Springer, 13-14 Dec. 2010. DOI: 10.1007/978-3-642-18009-5_9. [cited on page 211]
80. F. Chung and W. Zhao. A sharp PageRank algorithm with applications to edge ranking and graph sparsification. In R. Kumar and D. Sivakumar, editors, *Proc.*

of 7th Int. Workshop on Algorithms and Models for the Web-Graph, volume 6516 of *WAW 2010*, pages 2–14. Springer, Stanford, CA, USA, 13-14 Dec. 2010. DOI: 10.1007/978-3-642-18009-5_2. [cited on page 14, 265, 266]

81. A. Ciaramella, S. Cocozza, F. Iorio, G. Miele, F. Napolitano, M. Pinelli, G. Raiconi, and R. Tagliaferri. Interactive data analysis and clustering of genomic data. *Neural Networks*, 21, Issues 2-3:368–378, March-April 2008. [cited on page 269]
82. K.L. Clarkson. Nearest-neighbor searching and metric space dimensions. In G. Shakhnarovich, T. Darrell, and P. Indyk, editors, *Nearest-Neighbor Methods in Learning and Vision. Theory and Practice*, pages 26–71. MIT Press, Mar. 2006. [cited on page 79]
83. R. Coifman and S. Lafon. Diffusion maps. *Applied and Computational Harmonic Analysis*, 21:5–30, 2006. [cited on page 19, 144, 223, 224]
84. P. Corsini, B. Lazzerini, and F Marcelloni. A new fuzzy relational clustering algorithm based on the fuzzy *c*-means algorithm. *Soft Computing*, 9:439–447, 2005. [cited on page 123, 124]
85. D. Cvetković, P. Rowlinson, and S. Simić. Eigenvalue bounds for the signless Laplacian. *Publications de l'Institut Mathématique (Beograd)*, 81(95):11–27, 2007. DOI: 102298/PIM0795011C. [cited on page 181]
86. J. Czekanowski. *Zarys metod statystycznych w zastosowaniu do antropologii (An outline of statistical methods applied in anthropology)*, volume 5 of *Travaux de la Société des Sciences de Varsovie. III - Classe des sciences mathématiques et naturelles.* Towarzystwo Naukowe Warszawskie, Warszawa, 1913. reprint availabnle at `http://rcin.org.pl/dlibra/doccontent?id=29411&from=FBC`. [cited on page 22, 145]
87. S. Dasgupta and L. Schulman. A probabilistic analysis of EM for mixtures of separated, spherical Gaussians. *J. Machine Learning Research*, 8:203–226,, Feb. 2007. [cited on page 65, 66, 96, 97]
88. R.M. Dave. Fuzzy shell-clustering and applications to circle detection in digital images. *Int. J. General Systems*, 16(4):343–355, 1990. DOI: 10.1080/03081079008935087. [cited on page 115]
89. R.N. Dave. Characterization and detection of noise in clustering. *Pattern Recognition Letters*, 12(11):657–664, 1991. [cited on page 120]
90. R.N. Dave. Generalized fuzzy *c*-shells clustering and detection of circular and elliptical boundaries. *Pattern Recognition*, 25(7):713–721, Jul. 1992. DOI: 10.1016/0031-3203(92)90134-5. [cited on page 115]
91. R.N Dave and S. Sen. Robust fuzzy clustering of relational data. *IEEE Trans. Fuzzy Systems*, 10(6):713–727, Dec. 2001. [cited on page 51]
92. N.M.M. de Abreu. Old and new results on algebraic connectivity of graphs. *Linear Algebra and its Applications*, 423(1):53–73, 1 May 2007. DOI: 10.1016/j.laa.2006.08.017. [cited on page 248]
93. J.V. de Oliveira and W. Pedrycz, editors. *Advances in Fuzzy Clustering and its Applications*. J. Wiley & Sons, Ltd, New York, 2007. [cited on page 21, 109, 111]
94. A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. Royal Statistical Soc., Series B,*, 39:1–38, 1967. [cited on page 95, 97]
95. K Devarajan. Nonnegative matrix factorization: An analytical and interpretive tool in computational biology. *PLoS Comput. Biol.*, 4(7):e1000029, Jul. 2008. DOI: 10.1371/journal.pcbi.1000029. [cited on page 46]

96. I. S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the 7-th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD)*, San Francisco, California, USA, 26-29 Aug. 2001. [cited on page 152, 173, 182]
97. I.S. Dhillon, J. Fan, and Y. Guan. Efficient clustering of very large document collections. In R. Grossman, G. Kamath, and R. Naburu, editors, *Data Mining for Scientific and Engineering Applications*. Kluwer Academic Publishers, 2001. [cited on page 84]
98. I.S. Dhillon, Y. Guan, and J. Kogan. Iterative clustering of high dimensional text data augmented by local search. In *Proc. IEEE Int. Conf. Data Mining*, pages 131–138, Maebashi City, Japan, 2002. [cited on page 82, 83]
99. I.S. Dhillon, Y. Guan, and J. Kogan. Refining clusters in high dimensional text data. In *2-nd SIAM ICDM, Workshop on clustering high dimensional data*, Arlington, VA, 2002. http://citeseer.ist.psu.edu/dhillon02refining.html. [cited on page 82, 83]
100. I.S. Dhillon, Y. Guan, and B. Kulis. A unified view of kernel k-means, spectral clustering and graph cuts. Technical Report TR-04-25, University of Texas, Dept. of Computer Science, 2005. URL: `http://www.cs.utexas.edu/ftp/pub/techreports/tr04-25.pdf`. [cited on page 46]
101. I.S. Dhillon, Y. Guan, and B. Kulis. Weighted graph cuts without eigenvectors: A multilevel approach. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 29(11):1944–1957, Nov. 2007. DOI: 10.1109/TPAMI.2007.1115. [cited on page 146, 173, 203]
102. I.S. Dhillon and D.S. Modha. Concept decompositions for large sparse text data using clustering. *Machine Learning*, 42:143–175, 2001. [cited on page 82, 84]
103. I.S. Dhillon and S. Sra. Generalized nonnegative matrix approximations with Bregman divergences. In *Proc. of the Neural Information Processing Systems Conf.*, NIPS 2005, pages 283–290, Vancouver, Canada, Dec. 2005. [cited on page 46]
104. E. Dimitriadou, S. Dolničar, and A. Weingessel. An examination of indexes for determining the number of clusters in binary data sets. *Psychometrica*, 67(1):137–159, 2002. [cited on page 132]
105. C. Ding, X. He, H. Zha, M. Gu, and H. Simon. A min-max cut algorithm for graph partitioning and data clustering. In *Proc. IEEE Int. Conf. on Data Mining*, ICDM 2001, pages 107–114, San Jose, CA , USA, 10 Nov. - 2 Dec. 2001. DOI: 10.1109/ICDM.2001.989507. [cited on page 161]
106. Ch. Ding, X. He, and H.D. Simon. On the equivalence of nonnegative matrix factorization and spectral clustering. In *Proc. 5-th SIAM Intnl Conf. on Data Mining*, volume 5, pages 606–610, 2005. [cited on page 167]
107. C. Domeniconi, D. Gunopulos, S. Ma, B. Yan, M. Al-Razgan, and D. Papadopoulos. Locally adaptive metrics for clustering high dimensional data. *Data Mining and Knowledge Discovery*, 14(1):63–97, 2007. [cited on page 114]
108. W.E. Donath and A.J. Hoffman. Lower bounds for the partitioning of graphs. *IBM J. Res. Develop.*, 17:420–425, 1973. [cited on page 145]
109. B. Dorow, D. Widdows, K. Ling, J.-P. Eckmann, D. Sergi, and E. Moses. Using curvature and Markov clustering in graphs for lexical acquisition and word sense discrimination. In *Workshop MEANING-2005*, 2004. [cited on page 201]
110. P.G. Doyle and J.L. Snell. Random walks and electric networks. `arXiv:math/0001057v1 [math.PR]`, 11 Jan 2000. [cited on page 195, 197, 199, 253]
111. L. Duana, L. Xub, F. Guoc, J. Leea, and B. Yana. A local-density based spatial clustering algorithm with noise. *Information Systems*, 32(7):978–986, 2007. [cited on page 56]

112. R.O. Duda, P.E. Hart, and G. Stork. *Pattern Classification*. J. Wiley & Sons, New York, 2nd edition, 2000. [cited on page 18, 21, 73, 128]
113. D. Dueck. *Clustering by Affinity Propagation*. PhD thesis, Department of Electrical & Computer Engineering, University of Toronto, 2009. url: `http://www.cs.columbia.edu/~delbert/docs/DDueck-thesis_small.pdf`. [cited on page 125]
114. J.C. Dunn. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. *J. Cyber.*, 3(3):32–57, 1974. [cited on page 98]
115. M. Dyer and A Frieze. A simple heuristic for the p-center problem. *Oper. Res. Lett.*, 3(6):285–288, 1985. [cited on page 132]
116. W. E, T. Li, and E. Vanden-Eijnden. Optimal partition and effective dynamics of complex networks. *Proc. Nat. Acad. Sci. USA*, 105(23):7907–7912, 10 Jun. 2008. [cited on page 193]
117. Ch. Elkan. Using the triangle inequality to accelerate k-means. In T. Fawcett and N. Mishra, editors, *Proc. of the 20th Int. Conf. on Machine Learning*, ICML 2003, pages 147–153, Washington, DC, 21-24 Aug. 2003. AAAI Press. [cited on page 79]
118. A.J. Enright, S. van Dongen, and C.A. Ouzounis. An efficient algorithm for large-scale detection of protein families. *Nucleic Acids Research*, 30(7):1575–1584, 1 Apr. 2002. DOI: 10.1093/nar/30.7.1575. [cited on page 201]
119. A. Ester, H.-P. Kriegel, J. Sander, M. Wimmer, and X. Xu. Incremental clustering for mining in a data warehousing environment. In E. Simoudis, J. Han, and U.M. Fayyad, editors, *Proc. of the 2nd Int. Conf. on Knowledge Discovery and Data Mining*, pages 323–333, New York City, Aug. 1998. Morgan Kaufmann. [cited on page 54]
120. A. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large databases with noise. In *Proc. of the 24th VLDB Conf. on Knowledge Discovery and Data Mining*, pages 226–231, Portland, Oregon, Aug. 1996. AAAI Press. [cited on page 54]
121. B.S. Everitt. *Cluster Analysis*. Halsted Press, 1993. [cited on page 18, 21, 41]
122. M. Fiedler. Algebraic connectivity of graphs. *Czechoslovak Math. J.*, 23(98):298–305, 1973. [cited on page 145, 248]
123. M. Fiedler. A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory. *Czechoslovak Math. J.*, 25:619–672, 1975. [cited on page 145, 154, 248]
124. M. Filippone, F. Camastra, F. Masulli, and S. Rovetta. A survey on spectral and kernel methods for clustering. *Pattern Recognition*, 41(1):176–190, Jan. 2008. [cited on page 21, 53, 57, 59, 60]
125. M. Filippone, F. Masulli, and S. Rovetta. Applying the possibilistic c-means algorithm in kernel-induced spaces. *IEEE Trans. on Fuzzy Systems*, 18(3):572–584, Jun. 2010. DOI: 10.1109/TFUZZ.2010.2043440. [cited on page 122]
126. I. Fisher and J. Poland. Amplifying the block matrix structure for spectral clustering. In M. van Otterlo, M. Poel, and A. Nijholt, editors, *Proc. of the 14th Annual Machine Learning Conf. of Belgium and the Netherlands*, pages 21–28, 2005. `http://www.dr-fischer.org/pub/blockamp/index.html`. [cited on page 114, 176, 187, 189]
127. P. Fjällström. Algorithms for graph partitioning: A survey. *Linköping Electronic Articles in Computer and Information Science*, 3(10), 1998. [cited on page 53, 144]
128. E. Forgy. Cluster analysis of multivariate data: Efficiency versus interpretability of classification. *Biometrics*, pages 768–780, 1965. [cited on page 73, 75]
129. S. Fortunato. Community detection in graphs. *Physics Reports*, 486(3-5):75–174, 17 Feb. 2010. [cited on page 140, 205]

130. F. Fouss, A. Pirotte, J.-M. Renders, and M. Saerens. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE Trans. on Knowledge and Data Engineering*, 19(3):355–369, Mar. 2007. [cited on page 191, 197, 198, 250, 256, 257]
131. C. Fraley and A. Raftery. MCLUST: Software for model-based cluster and discriminant analysis. Technical Report 342, Dept. Statistics, University of Washington, 1999. [cited on page 96]
132. A.L.N. Fred and A.K. Jain. Data clustering using evidence accumulation. In *Proc. of the 16th Internat. Conf. on Pattern Recognition*, volume 4 of *ICPR 2002*, pages 276 – 280, DOI: 10.1109/ICPR.2002.1047450, 11-15 Aug. 2002. [cited on page 61]
133. A.L.N. Fred and A.K. Jain. Robust data clustering. In *Proc. of IEEE Computer Society Conf. on Computer Vision and Pattern Recognition*, volume 2 of *CVPR 2003*, pages 128–136, Madison, Wisconsin, USA, 16-22 Jun. 2003. [cited on page 61, 62, 139]
134. A.L.N. Fred and A.K. Jain. Combining multiple clusterings using evidence accumulation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 27(6):835–850, Jun. 2005. DOI: 10.1109/TPAMI.2005.113. [cited on page 62]
135. B.J. Frey and D. Dueck. Clustering by passing messages between data points. *Science*, 315:972–976, 16 Feb. 2007. [cited on page 13, 124, 125, 126]
136. B. Fritzke. Some competitive learning methods. Draft, Institute for Neural Computation, Ruhr-Universität Bochum, Germany, 5 Apr. 1997. URL: `http://www.ki.inf.tu-dresden.de/~fritzke/JavaPaper/`. [cited on page 81]
137. Y. Fukuyama and M. Sugeno. A new method of choosing the number of clusters for the fuzzy *c*-means method. In *Proc. 5th Fuzzy Syst. Symp.*, pages 247–250, 1989. (in Japanese). [cited on page 134]
138. C. Fyfe and J. Corchado. A comparison of kernel methods for instantiating case based reasoning systems. *Advanced Engineering Informatics*, 16(3):165–178, Jul. 2002. [cited on page 88]
139. I. Gath and A.B. Geva. Unsupervised optimal fuzzy clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7):773–781, 1989. [cited on page 109, 134]
140. A. Gionis, H. Mannila, and P. Tsaparas. Clustering aggregation. *ACM Trans. Knowl. Discov. Data*, 1(1), Mar. 2007. DOI: 10.1145/1217299.1217303. [cited on page 60]
141. M. Girvan and M.E.J. Newman. Community structure in social and biological networks. *Proc. Natl. Acad. Sci. USA*, 99(12):7821–7826, 11 Jun. 2002. DOI: 10.1073/pnas.122653799. [cited on page 205]
142. X. Golay, S. Kollias, G. Stoll, D. Meier, A. Valavanis, and P. Boesiger. A new correlation-based fuzzy logic clustering algorithm for fmri. *Magnetic Resonance in Medicine*, 40(2):249–260, Aug. 1998. DOI: 10.1002/mrm.1910400211. [cited on page 33]
143. G.H. Golub and Ch. F. van Loan. *Matrix Computation. Third edition.* Hindustan Book Agency, New Delhi, India, 1996. [cited on page 219, 238]
144. R.C. Gonzalez and R.E. Woods. *Digital Image Processing.* Prentice Hall, Upper Saddle River, NJ, third edition, 2008. [cited on page 115]
145. T. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306, 1985. [cited on page 75]
146. L. Grady. Random walks for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(11):1768–1783, Nov. 2006. [cited on page 19]

147. D. Graves and W. Pedrycz. Kernel-based fuzzy clustering and fuzzy clustering: A comparative experimental study. *Fuzzy Sets and Systems*, 161(4):522–543, 2010. [cited on page 105, 117, 118, 119]
148. S. Guattery and G.L. Miller. On the quality of spectral separators. *SIAM J. of Matrix Anal. Appl.*, 19(3):701–719, Jul. 1998. DOI: 10.1137/S0895479896312262. [cited on page 8, 170]
149. E.E. Gustafson and W.C. Kessel. Fuzzy clustering with a fuzzy covariance matrix. In *IEEE CDC*, pages 761–766, San Diego, CA, 1979. [cited on page 111]
150. I. Gutman and W. Xiao. Generalized inverse of the Laplacian matrix and some apptications. *Bull. de l'Academie Serbe des Sciences at des Artes (Cl. Math. Ntur.)*, 129:15–23, 2004. [cited on page 249]
151. A. Guttman. R-trees: a dynamic index structure for spatial searching. In *Proc. Third ACM SIGMOD Intnl. Conf. on Management of Data*, SIGMOD '84, pages 47–57, 1984. [cited on page 79]
152. I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *J. of Mach. Learn. Res.*, 3:1157–1182, Mar. 2003. [cited on page 128]
153. I. Guyon, S. Gunn, M. Nikravesh, and L.A. Zadeh, editors. *Feature Extraction. Foundations and Applications*, volume 207 of *Studies in Fuzziness and Soft Computing*. Springer, 2006. [cited on page 127]
154. L. Hagen and A. Kahng. New spectral methods for ratio cut partitioning and clustering. *IEEE Trans. on Computer-Aided Design*, 11(9):1074–1085, Sep. 1992. DOI: 10.1109/43.159993. [cited on page 156, 161, 173]
155. M. Halkidi, Y. Batistakis, and M. Vazirgiannis. On clustering validation techniques. *Intelligent Information Systems*, 17(2-3):107–145, 2001. [cited on page 132]
156. M. Halkidi and M. Vazirgiannis. Clustering validity assignment: Finding the optimal partitioning of a data set. In *Proc. ICDM Conf.*, 2001. [cited on page 132]
157. K.M. Hall. An r-dimensional quadratic placement algorithm. *Management Science*, 17(3):219–229, Nov. 1970. DOI: 10.1287/mnsc.17.3.219. [cited on page 146, 153, 156, 158, 160]
158. L.O. Hall, I.B. Ozyurt, and J.C. Bezdek. Clustering with a genetically optimized approach. *IEEE Trans. on Evolutionary Computation*, 3(2):103–112, Jul. 1999. DOI: 10.1109/4235.771164. [cited on page 108]
159. G. Hamerly. *Learnig structure and concepts in data using data clustering.* PhD thesis, University of California, San Diego, 2003. [cited on page 31, 97]
160. G. Hamerly and C. Elkan. Alternatives to the k-means algorithm that find better clusterings. In *Proc. of the ACM Conf. on Information and Knowledge Management*, CIKM-2002, pages 600–607, 2002. [cited on page 47, 86, 87]
161. J. Handl, J. Knowles, and D. B. Kell. Computational cluster validation in post-genomic data analysis. *Bioinformatics*, 21(15):3201–3212, Aug. 2005. DOI: 10.1093/bioinformatics/bti517. [cited on page 132]
162. P. Hansen and B. Jaumard. Cluster analysis and mathematical programming. *Matematical Programming*, 79(1-3):191–215, 1997. DOI: 10.1007/BF02614317. [cited on page 39, 42]
163. P. Hansen and N. Mladenović. j-Means: a new local search heuristic for minimum sum-of-squares clustering. *Pattern Recognition*, 34:405–413, 2002. [cited on page 73, 74]
164. T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning. Data Mining, Inference, and Prediction.* Springer Series in Statistics. Springer-Verlag Berlin, Heidelberg, New York, 2-nd edition, Feb. 2009. [cited on page 17, 90, 128, 222, 223]

165. R.J. Hathaway and J.C. Bezdek. Recent convergence results for the fuzzy c-means clustering algorithm. *J. of Classification*, 5:237–247, 1988. [cited on page 103, 104]
166. R.J. Hathaway and J.C. Bezdek. Optimization of clustering criteria by reformulation. *IEEE Trans. Fuzzy Systems*, 3:241–245, 1995. [cited on page 107, 108]
167. R.J. Hathaway and J.C. Bezdek. Generalized fuzzy c-means clustering strategies using L_p norm distances. *IEEE Trans. Fuzzy Systems*, 8(5):576–582, 2000. [cited on page 109, 111]
168. R.J. Hathaway, J.C Bezdek, and J.W. Davenport. On relational data versions of c-means algorithms. *Pattern Recognition Letters*, 17:607–612, 1996. [cited on page 51, 123]
169. R.J. Hathaway, J.C. Bezdek, and W. Tucker. An improved convergence theorem for the fuzzy c-means clustering algorithm. In J.C. Bezdek, editor, *The Analysis of Fuzzy Information*, volume 3, pages 1–10. CRC Press, Boca Raton, 1986. [cited on page 104]
170. R.J. Hathaway, J.W. Davenport, and J.C Bezdek. Relational duals of the c-means clustering algorithms. *Pattern Recognition*, 22:205–212, 1989. [cited on page 122]
171. T.C. Havens, R. Chitta, A.K. Jain, and R. Jin. Speedup of fuzzy and possibilistic kernel c-means for large-scale clustering. In *IEEE Int. Conf. on Fuzzy Systems*, pages 463 – 470, Grand Hyatt Taipei, Taipei, Taiwan, Jun. 27-30 2011. IEEE. DOI: 10.1109/FUZZY.2011.6007618. [cited on page 119]
172. X. He, H. Zha, Ch.H.Q. Ding, and H.D. Simon. Web document clustering using hyperlink structures. *Computational Statistics & Data Analysis*, 41(1):19–45, 28 Nov. 2002. DOI: 10.1016/S0167-9473(02)00070-1. [cited on page 147]
173. B. Hendrickson. Latent semantic analysis and Fiedler retrieval. *Linear Algebra and its Applications*, 421(2-3):345–355, 2007. DOI: 10.1016/j.laa.2006.09.026. [cited on page 147]
174. D.J. Higham, G. Kalna, and M. Kibble. Spectral clustering and its use in bioinformatics. *J. of Computational and Applied Mathematics*, 204(1):25–37, Jul. 2007. DOI: 10.1016/j.cam.2006.04.026. [cited on page 53, 147]
175. A. Hinneburg, E. Hinneburg, and D.A. Keim. An efficient approach to clustering in large multimedia databases with noise. In *Proc. of the 4th Int. Conf. on Knowledge Discovery and Datamining*, KDD 1998, pages 58–65, New York, NY, Sep. 1998. AAAI Press. [cited on page 55]
176. D.S. Hochbaum and D.B. Shmoys. A best possible heuristic for the k-center problem. *Mathematics of Operations Research*, 10(2):180–184, May 1985. [cited on page 75]
177. F. Höppner, F. Klawonn, R. Kruse, and T. Runkler. *Fuzzy Cluster Analysis*. J. Wiley & Sons, Chichester, England, 1999. [cited on page 109]
178. P. Hore, L.O. Hall, and D.B. Goldgof. Single pass fuzzy c means. In *IEEE Int. Conf. on Fuzzy Systems, FUZZ-IEEE*, Imperial College, London, UK, 2007. IEEE. URL: `http://www.csee.usf.edu/~hall/papers/singlepass.pdf`. [cited on page 109]
179. P. Hore, L.O. Hall, and D.B. Goldgof. A scalable framework for cluster ensembles. *Pattern Recognition*, 42(5):676–688, May 2009. DOI: 10.1016/j.patcog.2008.09.027. [cited on page 60, 62]
180. J. Hu, B.K. Ray, and M. Singh. Statistical methods for automated generation of service engagement staffing plans. *IBM J. Res. Dev.*, 51(3):281–293, 2007. [cited on page 17]
181. A. Huang. Similarity measures for text document clustering. In J. Holland, A. Nicholas, and D. Brignoli, editors, *Proc. New Zealand Comput. Sci. Res. Student Conf.*, NZCSRSC 2008, pages 49–56. Christchurch, New Zealand, 14-18 Apr.

2008. URL: `http://nzcsrsc08.canterbury.ac.nz/site/digital-proceedings`. [cited on page 32]

182. J.Z. Huang, M.K. Ng, H. Rong, and Z. Li. Automated variable weighting in k-means type clustering. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 27(5):657–668, May 2005. DOI: 10.1109/TPAMI.2005.95. [cited on page 128]
183. Z. Huang. Extensions to the k-means algorithm for clustering large data sets with categorical values. *Data Mining and Knowledge Discovery*, 2(3):283–304, Sep. 1998. DOI: 10.1023/A:1009769707641. [cited on page 13, 91, 92, 93, 94]
184. Z. Huang, A. Zhou, and G. Zhang. Non-negative matrix factorization: A short survey on methods and applications. springer berlin heidelberg, 2012. . In Z. Li, X. Li, Y. Liu, and Z. Cai, editors, *Computational Intelligence and Intelligent Systems*, volume 316 of *CCIS*, pages 331–340. Springer, 2012. [cited on page 46]
185. L. Hubert and Ph. Arabie. Comparing partitions. *J. of Classification*, 2(1):193–218, 1985. DOI: 10.1007/BF01908075. [cited on page 136]
186. T. Huntsberger and P. Ajjimarangsee. Parallel self-organizing feature maps for unsupervised pattern recognition. *Int. J. Gen. Syst.*, 16(4):357–372, May 1989. [cited on page 109]
187. H.M. Hussain, K. Benkrid, A. Ebrahim, A.T. Erdogan, and H. Seker. Novel dynamic partial reconfiguration implementation of k-means clustering on FPGAs: Comparative results with GPPs and GPUs. *Int. J. Reconfig. Comp.*, 2012. Article ID 135926. DOI: 10.1155/2012/135926. [cited on page 80]
188. M. Iosifescu. *Skończone procesy Markowa i ich zastosowania*. PWN, Warszawa, 1988. [cited on page 196, 199, 253, 263, 264]
189. M.A. Ismail and S.Z. Selim. Fuzzy c-means optimality of solutions and effective termination of the algorithm. *Pattern Recognition*, 19(6):481–485, 1986. [cited on page 100, 104]
190. A. Jain. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31:651–666, 2010. [cited on page 17, 21, 25, 69]
191. A. Jain and R. Dubes. *Algorithms for Clustering Data*. Prentice Hall, New Jersey, 1988. [cited on page 17, 18, 21, 91, 132]
192. A. Jain, M. Murty, and P. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31:264–323, 1999. [cited on page 21, 25, 132]
193. H. Jia, S. Ding, X. Xu, and R. Nie. The latest research progress on spectral clustering. *Neural Comput. & Applic.*, 24(7-8):1477–1486, Jun 2014. DOI: 10.1007/s00521-013-1439-2. [cited on page 147, 148, 179, 187]
194. R. Kannan, H. Salmasian, and S. Vempala. The spectral method for general mixture models. *SIAM J. Comput.*, 38(3):1141–1156, Jun. 2008. DOI: 10.1137/S0097539704445925. [cited on page 97]
195. R. Kannan, S. Vempala, and Vetta. On clusterings: Good, bad and spectral. *J. ACM*, 51(3):497–515, May 2004. [cited on page 132, 193]
196. T. Kanungo, D.M. Mount, N.S. Netanyahu, Ch.D. Piatko, R. Silverman, and A.Y. Wu. A local search approximation algorithm for k-means clustering. *Computational Geometry*, 28(2-3):89–112, Jun. 2004. DOI: 10.1016/j.comgeo.2004.03.003. [cited on page 72]
197. T. Kanungo, N.S. Netanyahu, Ch.D. Piatko, R. Silverman, and A.Y. Wu. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24(7):881–892, Jul. 2002. [cited on page 66, 78, 79]
198. B. Karrer, E. Levina, and M.E.J. Newman. Robustness of community structure in networks. *Phys. Rev. E*, 77(4):046119, 13 Apr. 2008. DOI: 10.1103/PhysRevE.77.046119. [cited on page 140]

199. G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J. on Scientific Computing*, 20(1):359–392, Aug. 1999. DOI: 10.1137/S1064827595287997. [cited on page 203]

200. I. Katsavounidis, C.-C.J. Kuo, and Z. Zhang. A new initialization technique for generalized Lloyd iteration. *IEEE Signal Processing Letters*, 1(10):144–146, Oct. 1994. DOI: 10.1109/97.329844. [cited on page 75]

201. L. Kaufman and P. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis.* John Wiley & Sons, New York, 1990. [cited on page 21, 76, 90, 133]

202. M. Kearns, Y. Mansour, and A.Y. Ng. An information-theoretic analysis of hard and soft assignment methods for clusterig. In *Proc. 13-th Conf. Uncertainty in Artificial Intelligence*, Madison WI, 1997. Morgan Kaufmann. [cited on page 97]

203. D.W. Kim, K.Y. Lee, D. Lee, and K.H. Lee. Evaluation of the performance of clustering algorithms in kernel-induced feature space. *Pattern Recognition*, 38(4):607–611, Apr. 2005. DOI: 10.1016/j.patcog.2004.09.006. [cited on page 59, 88]

204. T. Kim, J.C. Bezdek, and R.J. Hathaway. Optimality tests for fixed points of the fuzzy c-means algorithm. *Pattern Recognition*, 21:651–663, 1988. [cited on page 104]

205. J. Kleinberg. Hubs, authorities, and communities. *ACM Computing Surveys*, 31(4):5, Dec. 1999. DOI: 10.1145/345966.345982. [cited on page 173, 222]

206. J. Kleinberg. An impossibility theorem for clustering. In *Proc. NIPS 2002*, pages 446–453, 2002. http://books.nips.cc/papers/files/nips15/LT17.pdf. [cited on page 267]

207. Ph. A. Knight and D. Ruiz. A fast algorithm for matrix balancing. In A. Frommer, M.W. Mahoney, and D.B. Szyld, editors, *Web Information Retrieval and Linear Algebra Algorithms*, number 07071 in Dagstuhl Seminar Proceedings. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany, 2007. URL: `http://drops.dagstuhl.de/opus/volltexte/2007/1073`. [cited on page 216, 233]

208. Ph.A. Knight. The Sinkhorn-Knopp algorithm. convergence and applications. *SIMAX*, 30(1):261–275, 2008. [cited on page 45]

209. J. Kogan. *Introduction to Clustering Large and High-Dimensional Data.* Cambridge University Press, Cambridge, 2007. [cited on page 21, 85]

210. J.F. Kolen and T. Hutcheson. Reducing the time complexity of the fuzzy C-means algorithm. *IEEE Trans. Fuzzy Systems*, 10(2):263–267, 2002. [cited on page 109]

211. Y. Koren, R. Bell, and Ch. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, Aug. 2009. DOI: 10.1109/MC.2009.263. [cited on page 46]

212. Y. Koren, L. Carmel, and D. Harel. ACE: a fast multiscale eigenvectors computation for drawing huge graphs. In *Proc. of IEEE Symposium on Information Visualization*, INFOVIS 2002, pages 137–144, Boston, Massachusetts, USA, 29-29 Oct. 2002. IEEE Computer Society Washington, DC, USA. DOI: 10.1109/INFVIS.2002.1173159. [cited on page 240]

213. J. Koronacki and J. Ćwik. *Statystyczne systemy uczące się. 2-nd Edition.* EXIT, Warszawa, 2008. ISBN: 978-83-60434-56-7. [cited on page 17, 32, 39, 41, 60, 95]

214. H.-P. Kriegel, P. Krger, and A. Zimek. Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *ACM Tran. on Knowledge Discovery from Data (TKDD)*, 3(1):1, Mar. 2009. DOI: 10.1145/1497577.1497578. [cited on page 51]

215. R. Krishnapuram, A. Joshi, O. Nasraoui, and L. Yi. Low-complexity fuzzy relational clustering algorithms for web mining. *IEEE Transactions on Fuzzy Systems*, 9(4):595–608, 2001. [cited on page 124]

216. R. Krishnapuram and J.M. Keller. A possibilistic approach to clustering. *IEEE Trans. on Fuzzy Systems*, 2(2):98–110, May 1993. DOI: 10.1109/91.227387. [cited on page 120, 121]

217. R. Kruse, Ch. Döring, and M.-J. Lesot. Fundamentals of fuzzy clustering. In J. Valente de Oliveira and W. Pedrycz, editors, *Advances in Fuzzy Clustering and its Applications*, chapter 1, pages 3–30. J. Wiley & Sons, Ltd, 2007. DOI: 10.1002/9780470061190.ch1. [cited on page 114]

218. B. Kulis, A.C. Surendran, and J.C. Platt. Fast low-rank semidefinite programming for embedding and clustering. In M. Meila and X. Shen, editors, *Proc. 11th Int. Conf. on Artificial Intelligence and Statistics*, AISTAT 2007, pages 235–242, San Juan, Puerto Rico, 21-24 Mar. 2007. URL: `http://jmlr.csail.mit.edu/proceedings/papers/v2/kulis07a/kulis07a.pdf`. [cited on page 45]

219. A. Kumar, Y. Sabharwal, and S. Sen. A simple linear time $(1 + \varepsilon)$-approximation algorithm for k-means clustering in any dimensions. In *Proc. of the 45th Annual IEEE Symposium on Foudations of Computer Science*, FOCS'04, pages 454–462. IEEE Computer Society, 2004. [cited on page 72]

220. M. Kumar and J.B. Orlin. Scale-invariantclustering with minimum volume ellipsoids. *Computers & Operations Research*, 35(4):1017–1029, Apr. 2006. DOI: 10.1016/j.cor.2006.07.001. [cited on page 49, 50]

221. R. Kumar. Recommendation systems: a probabilistic analysis. *J. of Computer and System Sciences*, 63(1):42–61, Aug. 2001. DOI: 10.1006/jcss.2001.1757. [cited on page 53]

222. L.I. Kuncheva and D.P. Vetrov. Evaluation of stability of k-means cluster ensembles with respect to random initialization. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 28(11):1798–1808, Nov. 2006. [cited on page 62, 76]

223. S. Lafon, Y. Keller, and R. R. Coifman. Data fusion and multi-cue data matching by diffusion maps. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(11):1784–1797, 2006. [cited on page 225]

224. S. Lafon and A.B. Lee. Diffusion maps and coarse-graining: A unified framework for dimensionality reduction, graph partitioning, and data set parameterization. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 28(9):1393–1403, Sep. 2006. [cited on page 193]

225. A. Lancichinetti, S. Fortunato, and J. Kertész. Detecting the overlapping and hierarchical community structure of complex networks. *New Journal of Physics*, 11:033015, 10 Mar. 2009. DOI: 10.1088/1367-2630/11/3/033015. [cited on page 139, 140, 141]

226. K. Lang. Fixing two weaknesses of the spectral method. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 715–722. MIT Press, Cambridge, MA, 2006. [cited on page 204]

227. G.F. Lawler. *Random Walk and the Heat Equation*, volume 55 of *Student Mathematical Library*. Providence, R.I.: American Mathematical Society, 2010. [cited on page 253]

228. D. Lee and H.S. Seung. Learning the parts of objects by nonnegative matrix factorization. *Nature*, 401(6755):788–791, 21 Oct. 1999. DOI: 10.1038/44565. [cited on page 46]

229. J.A. Lee and M. Verleysen. *Nonlinear Dimensionality Reduction*. Information Science and Statistics. Springer, 2007. [cited on page 148, 223]

230. T. Li. Clustering based on matrix approximation: A unifying view. *Knowledge & Information Systems*, 17(1):1–15, 2008. DOI: 10.1007/s10115-007-0116-0. [cited on page 46, 167]

231. T. Li and Ch. Ding. Non-negative matrix factorizations for clustering: A survey. In C.C. Aggarwal and C.K. Reddy, editors, *Data Clustering: Algorithms and Applications*, Data Mining and Knowledge Discovery Series, chapter 7, pages 149–176. Chapman & Hall/CRC, 2013. [cited on page 46, 167]
232. T.W. Liao. Clustering of time series data – a survey. *Pattern Recognition*, 38(11):1857–1874, Nov. 2005. DOI: 10.1016/j.patcog.2005.01.025. [cited on page 35]
233. Y. Lifshits. The homepage of nearest neighbors and similarity search, 2004-2007. `http://simsearch.yury.name/tutorial.html`. [cited on page 79]
234. F. Lin and W.C. Cohen. A very fast method for clustering big text datasets. In *Proc. of the 19th European Conf. on Artificial Intelligence*, ECAI 2010, pages 303–308, Amsterdam, The Netherlands, The Netherlands, 2010. IOS Press. [cited on page 13, 219, 221]
235. Y. Linde, A. Buzo, and R. (1980) 28: 84. Gray. An algorithm for vector quantizer design. *IEEE Trans. on Communications*, 28(1):84–95, Jan. 1980. DOI: 10.1109/TCOM.1980.1094577. [cited on page 69]
236. H. Liu and H. Motoda, editors. *Computational Methods of Feature Selection.* CRC Data Mining and Knowledge Discovery Series. Chapman and Hall/CRC, Bocca Raton, FL, 2007. [cited on page 127]
237. N. Liu. *Spectral clustering for graphs and Markov chains.* PhD thesis, North Carolina State University, Raleigh, North Carolina, USA, 2010. [cited on page 161, 182]
238. S.P. Lloyd. Least squares quantization in pcm. *IEEE Trans. on Information Theory*, 28(2):129–137, Mar. 1982. DOI: 10.1109/TIT.1982.1056489. [cited on page 47, 69]
239. L. Lovász. Random walks on graphs: a survey. *Combinatorics*, 2:353–398, 1993. [cited on page 253, 257, 258]
240. J. MacQueen. Some methods for classifications and analysis of multivariate observations. In *Proc. of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, Berkeley, 1967. University of California Press. [cited on page 69, 81]
241. K. Macropol, T. Can, and A.K Singh. RRW: repeated random walks on genome-scale protein networks for local cluster discovery. *BMC Bioinformatics*, 10(1):283–292, 2009. DOI: 10.1186/1471-2105-10-283. [cited on page 205, 211]
242. R. De Maesschalck, D. Jouan-Rimbaud, and D.L. Massart. The Mahalanobis distance. *Chemometrics and Intelligent Laboratory Systems*, 50:1–18, 2000. [cited on page 30]
243. M. Maier, U. von Luxburg, and M. Hein. Influence of graph construction on graph based clustering measures. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing*, volume 21, pages 1025–1032. Curran, Red Hook, 2009. [cited on page 148]
244. R. Maitra, A.D. Peterson, and A.P. Ghosh. A systematic evaluation of different methods for initializing the K-means clustering algorithm. Preprint, 2010. `http://www.public.iastate.edu/~apghosh/files/IEEEclust2.pdf`. [cited on page 76]
245. M. Manguoglu. A highly efficient parallel algorithm for computing the Fiedler vector. `arXiv:1003.3689v1 [cs.NA]`, 18 Mar. 2010. [cited on page 146]
246. Ch.D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval.* Cambridge University Press, 1 Apr. 2009. [cited on page 17, 137, 217]
247. G.J. McLachlan and T. Krishnan. *The EM Algorithm and Extensions.* Wiley & Sons, 1997. [cited on page 95]

248. F. McSherry. *Spectral methods for data analysis*. PhD thesis, University of Washington, Seattle, WA, 2004. URL: `http://khup.com/download/2_keyword-matrix-of-spectral-data/spectral-methods-for-data-analysis.pdf`. [cited on page 147]
249. M. Meilă. Comparing clusterings – an information based distance. *J. of Multivariate Analysis*, 98(5):873–895, May 2007. [cited on page 136, 138, 140]
250. M. Meilă and D. Heckerman. An experimental comparison of model-base clustering methods. *Mach. Learning*, 42(1/2):9–29, Jan.-Feb. 2001. DOI: 10.1023/A:1007648401407. [cited on page 138]
251. M. Meilă and W. Pentney. Clustering by weighted cuts in directed graphs. In *Proc. of the 7th SIAM Int. Conf. on Data Mining*, Radisson University Hotel, Minneapolis, Minnesota, 26-28 Apr. 2007. [cited on page 173]
252. M. Meilă and J. Shi. A random walks view on spectral segmentation. In *8th Int. Workshop on AI and Statistics, AISTATS 2001*. Hyatt Hotel, Key West, Florida, USA, 4-7 Jan. 2001. URL: `http://www.gatsby.ucl.ac.uk/aistats/aistats2001/files/meila177.ps`. [cited on page 19, 175, 192, 193, 194, 220]
253. M. Meilă and L. Xu. Multiway cuts and spectral clustering. Technical report, University of Washington, 2003. [cited on page 175]
254. M.E.S. Mendes and L. Sacks. Dynamic knowledge representation for e-learning applications. In M. Nikravesh, B. Azvine, R. Yager, and L.A. Zadeh, editors, *Enhancing the Power of the Internet*, volume 139 of *Studies in Fuzziness and Soft Computing*, pages 259–282. Springer-Verlag Berlin, Heidelberg, Berlin Heidelberg New York, Jan. 2004. [cited on page 116]
255. C.D. Meyer. *Matrix Analysis and Applied Linear Algebra*. SIAM, Philadelphia, 2000. [cited on page 231, 235, 239, 242]
256. C.D. Meyer and Ch. D. Wessell. Stochastic data clustering. `arXiv:1008.1758v2 [math.NA]`, 25 Apr 2012. [cited on page 13, 214, 215, 216]
257. G. Milligan and M. Cooper. An examination of procedures for determining the number of clusters in data sets. *Psychometrica*, 50(2):159–179, Jun. 1985. [cited on page 132]
258. G. Mishne and I. Cohen. Multiscale anomaly detection using diffusion maps. *IEEE J. of Selected Topics in Signal Processing*, 7(1):111 – 123, Jan 2013. DOI: 10.1109/JSTSP.2012.2232279. [cited on page 225]
259. B. Mohar. The Laplacian spectrum of graphs. In Y. Alavi, G. Chartrand, O. R. Oellermann, and A.J. Schwenk, editors, *Proceedings of the Sixth Quadrennial Int. Conf. on the Theory and Applications of Graphs*, volume 2, pages 871–898, Kalamazoo, MI, 1991. Wiley, New York. [cited on page 145, 247]
260. B. Mohar. Some applications of Laplace eigenvalues of graphs. In G. Hahn and G. Sabidussi, editors, *Graph Symmetry: Algebraic Methods and Applications*, volume 497 of *NATO ASI Ser. C*, pages 225–275. Kluwer, 1997. [cited on page 247]
261. A. Moore. Very fast EM-based mixture model clustering using multiresolution kd-trees. In M. Kearns and D. Cohn, editors, *Proceedings of the 1998 Conf. on Advances in Neural Information Processing Systems II*, pages 543–549. Morgan Kaufman, Apr. 1998. [cited on page 97]
262. A. Moore. The anchors hierarchy: Using the triangle inequality to survive high-dimensional data. In *Proc. of the 12th Conf. on Uncertainty in Artificial Intelligence*, pages 397–405. AAAI Press, 2000. [cited on page 13, 57]
263. B. Nadler and M Galun. Fundamental limitations of spectral clustering. In B. Schölkopf, J. Platt, and Th. Hofmann, editors, *Proc. of the 2006 Conf. Advances in Neural Information Processing Systems*, number 19 in NIPS 2007, pages 1017–1024. MIT Press, 2007. [cited on page 190]

264. F. Napolitano, G. Raiconi, R. Tagliaferri, A. Ciaramella, A. Staiano, and G. Miele. Clustering and visualization approaches for human cell cycle gene ex- pression data analysis. *International Journal of Approximate Reasoning*, 47, Issue 1:70–84, January 2008. [cited on page 269]
265. M.C.V. Nascimento and A.C.P.L.F. de Carvalho. Spectral methods for graph clustering – a survey. *European J. of Op. Res.*, 211(2):221–231, Jun 2011. DOI: 10.1016/j.ejor.2010.08.012. [cited on page 147]
266. O. Nasraoui, M. Soliman, E. Saka, A. Badia, and R. Germain. A Web usage mining framework for mining evolving user profiles in dynamic websites. *IEEE Trans. on Knowledge and Data Engineering*, 20(2):202–215, Feb. 2008. [cited on page 124]
267. M.E.J. Newman. Detecting community structure in networks. *Eur. Phys. J. B*, 38:321–330, 25 Mar. 2004. DOI: 10.1140/epjb/e2004-00124-y. [cited on page 205]
268. M.E.J. Newman. Finding community structure in networks using the eigenvectors of matrices. *Phys. Rev. E*, 74(3):036104, 2006. DOI: 10.1103/PhysRevE.74.036104. [cited on page 146, 205]
269. A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: analysis and an algorithm. In T. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems*, volume 14, pages 849–856. MIT Press, 2002. [cited on page 13, 76, 175, 176, 177, 178, 179, 186, 187]
270. R. Ng and J. Han. Efficient and effective clustering methods for spatial data mining. In *Proc. 20-th Conf. on VLDB*, pages 144–155, Santiago, Chile, 1994. [cited on page 91]
271. C.K. Nicholas and R. Dahlberg. Spotting topics with the singular value decomposition. In *Proc. of the 4th Int. Workshop on Principles of Digital Document Processing*, volume 1481 of *LNCS*, pages 82–91. Springer-Verlag, 1998. [cited on page 173]
272. P. Orponen, S.E. Schaeffer, and V.A. Gaytán. Locally computable approximations for spectral clustering and absorption times of random walks. `arXiv:0810.4061v1 [cs.DM]`, 22 Oct. 2000. [cited on page 199, 200]
273. R. Ostrovsky, Y. Rabani, L.J. Schulman, and Ch. Swamy. The effectiveness of Lloyd-type methods for the k-means problem. In *Proc. of the 47th Annual IEEE Symposium on Foundations of Computer Science*, FOCS'06, pages 165–176, Berkeley, CA, 21-24 Oct. 2006. [cited on page 70, 77]
274. L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the Web. Technical report, Stanford Digital Library Technologies Project, 1998. URL: `citeseer.ist.psu.edu/page98pagerank.html`. [cited on page 19, 209, 259, 261]
275. N.R. Pal and J.C. Bezdek. On cluster validity for the fuzzy c-means model. *IEEE Trans. Fuzzy Syst.*, 3(3):370–379, 1995. [cited on page 102]
276. N.R. Pal, J.C. Bezdek, and E.C.-K. Tsao. Generalized clustering networks and Kohonen's self-organizing scheme. *IEEE Trans. Neural Networks*, 4(4):549–557, Jul. 1993. [cited on page 109]
277. N.R. Pal, K. Pal, J.M. Keller, and J.C. Bezdek. A possibilistic fuzzy c-means clustering algorithm. *IEEE Trans. on Fuzzy Systems*, 13(4):517–530, Aug. 2005. [cited on page 122]
278. H.-S. Park and C.-H. Jun. A simple and fast algorithm for k-medoids clustering. *Expert Syst. Appl.*, 36(2):3336–3341, Mar. 2009. DOI: 10.1016/j.eswa.2008.01.039. [cited on page 91]
279. L. Parsons, E. Haque, and H. Liu. Subspace clustering for high dimensional data: a review. *ACM SIGKDD Explorations Newsletter*, 6(1):90–105, Jun. 2004. DOI: 10.1145/1007730.1007731. [cited on page 51]

280. W. Pedrycz. *Knowledge-based Clustering.* Wiley, Hoboken, NJ, 2005. [cited on page 21, 104]
281. J.M. Pena, J.A. Lozano, and P. Larrañaga. An empirical comparison of four initialization methods for the k-means algorithm. *Pattern Recognition Letters*, 20:1027–1040, 1999. [cited on page 76]
282. J. Peng and Y. Wei. Approximating K-means-type clustering via semidefinite programming. *SIAM J. on Optimization*, 18(1):186–205, 2007. DOI: 10.1137/050641983. [cited on page 42, 44, 45, 49]
283. A. Pérez-Suárez, J.F. Martinéz-Trinidad, J.A. Carrasco-Ochoa, and J.E. Medina-Pagola. A new overlapping clustering algorithm based on graph theory. In *Advances in Artificial Intelligence*, number 7629 in LNCS, pages 61–72. Springer, 2013. [cited on page 51]
284. F. Pernkopf and D. Bouchaffra. Genetic-based EM algorithm for learning Gaussian mixture models. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 27(8):1344–1348, Aug. 1344-1348. DOI: 10.1109/TPAMI.2005.162. [cited on page 98]
285. A. Pothen, H.D. Simon, and K.-P. Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM J. Matrix Anal. Appl.*, 11(3):430–452, 1990. [cited on page 146, 173]
286. F. P. Preparata and M. I. Shamos. *Computational Geometry.* Springer-Verlag Berlin, Heidelberg, New York, 1990. [cited on page 48]
287. W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes: The Art of Scientific Computing, (third ed.).* Cambridge University Press, Cambridge, New York, 1992. [cited on page 108, 231]
288. H. Qiu and E.R. Hancock. Clustering and embedding using commute times. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 29(11):1873–1890, 2007. [cited on page 162, 197, 251]
289. A. Rajaraman and J.D. Ullman. *Mining of Massive Data Sets.* Stanford University, 2010. URL: `http://infolab.stanford.edu/~ullman/mmds.html`. [cited on page 21, 27, 130]
290. A. Ralston. *Wstęp do analizy numerycznej.* PWN, Warszawa, 1971. [cited on page 231, 239]
291. D. Rao, D. Yarowsky, and Ch. Callison-Burch. Affinity measures based on the graph laplacian. In *Proc. of the Third Textgraphs Workshop on Graph-Based Algorithms for Natural Language Processing*, TextGraphs-3, pages 41–48, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics. [cited on page 191]
292. N. Rebagliati and A. Verri. Spectral clustering with more than K eigenvectors. *Neurocomputing*, 74(9):1391–1401, Apr 2011. DOI: 10.1016/j.neurocom.2010.12.08. [cited on page 179]
293. R.A. Redner and H.F. Walker. Mixture densities, maximum likelihood and the EM algorithm. *SIAM Review*, 26(2):195–239, 1984. [cited on page 65]
294. J.L. Rodgers and W.A. Nicewander. Thirteen ways to look at the correlation coefficient. *The American Statistician, Vol. 42, No. 1 (Feb., 1988), pp. 59-66*, 42(1):59–66, Feb. 1988. url: `http://www.jstor.org/stable/2685263`. [cited on page 34]
295. M. Roubens. Pattern classification problems and fuzzy sets. *Fuzzy Sets and Systems*, 1:239–253, 1978. [cited on page 122]
296. S.T. Roweis and L.K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 22 Dec. 2000. DOI: 10.1126/science.290.5500.2323. [cited on page 223]

297. E.H. Ruspini. A new approach to clustering. *Inf. Control*, 15:22–32, 1969. [cited on page 98]
298. Y. Saad. *Numerical Methods for Large Eigenvalue Problems.* SIAM, Philadelphia, 2-nd edition, 2011. [cited on page 231, 238]
299. J. Sander, A. Ester, H.-P. Kriegel, and X Xu. Density-based clustering in spatial databases: The algorithm GDBSCAN and its applications. *Data Mining and Knowledge Discovery*, 2(2):169–194, 1998. [cited on page 54, 56]
300. B.M. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Recommender systems for large-scale e-commerce: Scalable neighborhood formation using clustering. In *Proc. of the Fifth Int. Conf. on Computer and Information Technology*, ICCIT 2002, 2002. [cited on page 17]
301. V. Satuluri and S. Parthasarathy. Scalable graph clustering using stochastic flows: applications to community discovery. In *Proc. of the 15th ACM SIGKDD Intnal Conf. on Knowledge Discovery and Data Mining*, KDD'09, pages 737–746, Paris, France, 28 Jun. - 01 Jul. 2009. ACM New York, NY, USA. DOI: 10.1145/1557019.1557101. [cited on page 201, 203]
302. V. Satuluri, S. Parthasarathy, and Y. Ruan. Local graph sparsification for scalable clustering. In *Int. Conf. on Management of Data, SIGMOD/PODS'11*, pages 721–732, Athens, Greece, 12-16 Jun. 2011. ACM New York, NY, USA. DOI: 10.1145/1989323.1989399. [cited on page 203]
303. V. Satuluri, S. Parthasarathy, and D. Ucar. Markov clustering of protein interaction networks with improved balance and scalability. In *Proc. of the First ACM Int. Conf. on Bioinformatics and Computational Biology*, pages 247–256, Niagara Falls, NY, USA, 2-4 Aug. 2010. ACM New York, NY, USA. DOI: 10.1145/1854776.1854812. [cited on page 201, 203]
304. L.K. Saul and S.T. Roweis. Think globally, fit locally: unsupervised learning of low dimensional manifolds. *J. of Machine Learning Research*, 4:119–155, 1 Dec. 2003. DOI: 10.1162/153244304322972667. [cited on page 114, 223]
305. S.E. Schaeffer. Graph clustering. *Computer Science Review*, 1(1):27–64, Aug. 2007. DOI: 10.1016/j.cosrev.2007.05.001. [cited on page 53, 144, 147]
306. B. Schölkopf, A.J. Smola, and K.R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Comput.*, 10(5):1299–1319, 1998. [cited on page 88]
307. M. Schonlau. The clustergram: a graph for visualizing hierarchical and non-hierarchical cluster analyses. *The Stata Journal*, 2(4):391–402, 2002. [cited on page 131]
308. V. Schwämmle and O.N. Jensen. A simple and fast method to determine the parameters for fuzzy c-means cluster validation. arxiv:1004.1307v1 [q-bio.qm], Department of Biochemistry and Molecular Biology, University of Southern Denmark, DK-5230 Odense M,Denmark, Apr. 2010. [cited on page 103]
309. A.J. Seary and W.D. Richards. Spectral methods for analyzing and visualizing networks: an introduction. In R.L. Breiger, K.M. Carley, and Ph. Pattison, editors, *Dynamic Social Network Modeling and Analysis: workshop summary and papers*, pages 209–228. National Academy Press, Washington, DC, 2003. [cited on page 160, 169]
310. S.Z. Selim and M.A. Ismail. k-means-type algorithms: a generalized convergence theorem and characterization of local optimality. *IEEE. Trans. on Pattern Analysis and Mach. Intell.*, 6:81–87, 1984. [cited on page 73]
311. J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, Aug. 2000. [cited on page 13, 146, 152, 161, 162, 164, 168, 172, 173, 174, 175, 193]

312. T. Shi, M. Belkin, and B. Yu. Data spectroscopy: Eigenspaces of convolution operators and clustering. *The Annals of Statistics*, 37(6B):3960–3984, 2009. DOI: 10.1214/09-AOS700. [cited on page 13, 172, 179, 180, 181, 194]

313. R. Shioda and L. Tunçel. Clustering via minimum volume ellipsoids. *J. Computational Optimization and Applications*, 37(3):247–295, Jul. 2007. DOI: 10.1007/s10589-007-9024-1. [cited on page 49]

314. L. Shu, A. Chen, M. Xiong, and W. Meng. Efficient spectral neighborhood blocking for entity resolution. In *Proc. IEEE Intn'l Conf on Data Engineering*, ICDE 2011, pages 1067–1078, Hannover, Germany, Apr. 2011. URL: `http://www.cs.binghamton.edu/~mrng/pub.d/ICDE11_conf_full_065_update.pdf`. [cited on page 217]

315. A. Singer, R. Erban, I.G. Kevrekidis, and R.R. Coifman. Detecting intrinsic slow variables in stochastic dynamical systems by anisotropic diffusion maps. *PNAS*, 106(38):16090–16095, 22 Sep. 2009. DOI: 10.1073/pnas.0905547106. [cited on page 187]

316. R. Sinkhorn. A relationship between arbitrary positive matrices and doubly stochastic matrices. *The Annals of Math. Stat.*, 35:876–879, 1964. [cited on page 233]

317. T. Sipola. *Knowledge Discovery Using Diffusion Maps.* PhD thesis, University of Jyvaskälä, Jyvaskälä, Finland, 2013. `https://jyx.jyu.fi/dspace/handle/123456789/42647?show=full`. [cited on page 225]

318. P.H. Sneath and R.R. Sokal. *Numerical Taxonomy.* Freeman, San Francisco, 1973. [cited on page 35]

319. H. Späth. *Cluster Analysis Algorithms for Data Reduction and Classification of Objects.* Ellis Harwood, Chichester, 1980. [cited on page 21, 73, 74]

320. D. Spielman. Spectral graph theory. In U. Naumann and O. Schenk, editors, *Combinatorial Scientific Computing*, chapter 16. Chapman & Hall/CRC Computational Science, 25 Jan. 2012. URL: `http://www.cs.yale.edu/~spielman/PAPERS/SGTChapter.pdf`. [cited on page 147, 157, 159]

321. D.A. Spielman and S.-H. Teng. Spectral partitioning works: planar graphs and finite element meshes. *Proc. 37th Annual IEEE Symposium on Foundations of Computer Science*, pages 96–105, 1996. DOI: 10.1109/SFCS.1996.548468. [cited on page 205]

322. D.A. Spielman and S.-H. Teng. Solving sparse, symmetric, diagonally-domiant linear systems in time $O(m^{1.31})$. In *Proc. 44th Annual IEEE Symposium on Foundations of Computer Science*, FOCS'03, pages 416–427, Los Alamitos, CA, USA, 11-14 Oct. 2003. IEEE Computer Society. DOI: 10.1109/SFCS.2003.1238215. [cited on page 212]

323. D.A. Spielman and S.-H. Teng. A local clustering algorithm for massive graphs and its application to nearly-linear time graph partitioning. `arXiv:0809.3232v1 [cs.DS]`, 18 Sep. 2008. [cited on page 13, 192, 205, 206, 207, 208]

324. N. Srebro, G. Shakhnarovich, and S. Roweis. When is clustering hard? PASCAL Workshop on Statistics and Optimization of Clustering Workshop, Jul. 2005. URL: `http://ttic.uchicago.edu/~nati/Publications/SrebroEtalPASCAL05.pdf`. [cited on page 65]

325. M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. In *Proc. of KDD Workshop on Text Mining, Proc. of the 6th Int. Conf. on Knowledge discovery and Data Mining*, Boston, MA, Aug. 2000. URL: `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.34.1505`. [cited on page 81, 82]

326. H. Steinhaus. Sur la division des corp materiels en parties. *Bull. Acad. Polon. Sci.*, 4(12):801–804, 1956. [cited on page 69]

327. A. Strehl and J. Ghosh. Cluster ensembles – a knowledge reuse framework for combining multiple partitions. *J. of Machine Learning Research*, 3:583–617, Dec. 2002. DOI: 10.1162/153244303321897735. [cited on page 60, 61, 62, 139]

328. T. Su and J.G. Dy. In search of deterministic methods for initializing k-means and Gaussian mixture clustering. *Intelligent Data Analysis*, 11(4):319–338, Sep. 2007. [cited on page 76]

329. C.A. Sugar and G.M. James. Finding the number of clusters in a data set: An information theoretic approach. *J. of the American Statistical Association*, 98(463):750–763, Sep. 2003. [cited on page 131]

330. P. Sun and R.M. Freund. Computation of minimum-volume covering ellipsoids. *Operations Research*, 52(5):690–706, 2004. DOI: 10.1287/opre.1040.0115. [cited on page 50]

331. A. Szlam and X. Bresson. Total variation and Cheeger cuts. In J. Fürnkranz and Th. Joachims, editors, *Proc. of the 27th Intl Conf. on Machine Learning (ICML-10)*, pages 1039–1046, Haifa, Israel, June 2010. Omnipress. [cited on page 161]

332. P.-N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining.* Addison Wesley Longman, 2006. [cited on page 21]

333. K. Thangavel and N.K. Visalakshi. Ensemble based distributed k-harmonic means clustering. *Int. J. of Recent Trends in Engineering*, 2(1):125–129, Nov. 2009. [cited on page 62]

334. R. Tibshirani, G. Walther, and T. Hastie. Estimating the number of clusters in a data set via the gap statistic. *J. of the Royal Stat. Soc.: Ser. B (Statistical Methodology)*, 63(2):411–423, Jan. 2001. DOI: 10.1111/1467-9868.00293. [cited on page 130]

335. D.A. Tolliver. *Spectral rounding & image segmentation.* PhD thesis, Carnegie Mellon University, Pittsburgh, PA, Aug. 2006. [cited on page 169, 192]

336. D.A. Tolliver and G.L. Miller. Graph partitioning by spectral rounding: Applications in image segmentation and clustering. In A. Fitzgibbon, C.J. Taylor, and Y. LeCun, editors, *IEEE Computer Society Conf. on Computer Vision and Pattern Recognition*, volume 1, pages 1053 – 1060. IEEE Computer Soc., 17-22 Jun. 2006. DOI: 10.1109/CVPR.2006.129. [cited on page 156, 167]

337. J.T. Tou and R.C. Gonzalez. *Pattern Recognition Principles.* Addison-Wesely Pub. Co., 1974. [cited on page 21, 57, 80]

338. L. N. Trefethen and D. Bau, III. *Numerical Linear Algebra.* SIAM, Philadelphia, Apr. 1997. [cited on page 231, 238]

339. M. Trosset. Visualizing correlation. *J. of Computational and Graphical Statistics*, 14(1):1–19, 2005. DOI: 10.1198/106186005X27004. [cited on page 33]

340. E.C.-K. Tsao, J.C. Bezdek, and N.R. Pal. Fuzzy Kohonen clustering networks. *Pattern Recognition*, 27(5):757–764, May 1994. [cited on page 109]

341. J.K. Uhlmann. Satisfying general proximity/similarity queries with metric trees. *Info. Processing Lett.*, 40:175–179, 1991. [cited on page 79]

342. L. G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, November 1984. [cited on page 271]

343. S. van Dongen. *Graph Clustering by Flow Simulation.* PhD thesis, University of Utrecht, Amsterdam, May 2000. [cited on page 201, 202]

344. S. van Dongen. Performance criteria for graph clustering and Markov cluster experiments. Technical Report INS-R0012, Centre for Mathematics and Computer Science, Amsterdam, The Netherlands, 31 May 2000. `http://www.cwi.nl/ftp/CWIreports/INS/INS-R0012.pdf`. [cited on page 136, 137]

345. S. van Dongen. Graph clustering via a discrete uncoupling process. *SIAM J. Matrix Analysis & Applications*, 30(1):121–141, 2008. [cited on page 201]
346. R. van Driessche and D. Roose. An improved spectral bisection algorithm and its application to dynamic load balancing. *Parallel Computing*, 21(1):29–48, Jan. 1995. [cited on page 145]
347. V.N. Vapnik. *The Nature of Statistical Learning Theory.* Springer, New York, 1995. [cited on page 57]
348. S. Vempala and G. Wang. A spectral algorithm for learning mixture models. *J. of Computer and System Sciences*, 68(4):841–860, Jun. 2004. DOI: 10.1016/j.jcss.2003.11.008. [cited on page 66]
349. D. Verma and M. Meilă. Comparison of spectral clustering methods. Technical Report TR CSE-03-05-01, Uniwersity of Washington, Seattle, WA, USA, 2003. [cited on page 13, 171, 174, 175]
350. N.K. Vishnoi. Laplacian solvers and their algorithmic applications. *Foundations and Trends®in Theoretical Computer Science*, 8(1-2):1–141, 2013. doi: 10.1561/0400000054. [cited on page 171]
351. U. von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007. [cited on page 53, 147, 148, 153, 161, 165, 166, 171, 185, 186, 249, 250]
352. U. von Luxburg, M. Belkin, and O. Bousquet. Consistency of spectral clustering. *The Annals of Statistics*, 36(2):555–586, 2008. DOI: 10.1214/009053607000000640. [cited on page 170]
353. U. von Luxburg, A. Radl, and M. Hein. Getting lost in space: Large sample analysis of the commute distance. In J. Lafferty, C.K.I. Williams, J. Shawe-Taylor, R.S. Zemel, and A. Culotta, editors, *Proc. of the 23rd Conf. Advances in Neural Information Processing Systems*, NIPS 2010, pages 2622–2630, Vancouver, British Columbia, Canada, 6-9 Dec. 2010. Curran Associates, Inc. [cited on page 198]
354. S. Wagner and D. Wagner. Comparing clustering. Interner Bericht. Fakultät für Informatik, Universität Karlsruhe, 12 Jan. 2007. `http://digbib.ubka.uni-karlsruhe.de/volltexte/1000011477`. [cited on page 135, 136, 137, 138]
355. K. Wagstaff, C. Cardie, S. Rogers, and S. Schroedl. Constrained k-means clustering with background knowledge. In *Proc. of the 18-th Int. Conf. on Machine Learning*, pages 577–584, Williamstown, MA, USA, 28 Jun. - 1 Jul. 2001. [cited on page 80]
356. D.L. Wallace. Comment. *J. of the Americal Statistical Association*, 78(383):569–576, 1983. [cited on page 138]
357. C. Walles and D. Dowe. Intristic classification by MML – the Snob program. In *Proc. 7th Australian Joint Conf. on Artificial Intelligence*, pages 37–44, Armidale, Australia, 1994. World Scientific Publishing Co. [cited on page 96]
358. F. Wang, P. Li, A.Ch. König, and M. Wan. Improving clustering by learning a bi-stochastic data similarity matrix. *Knowl. Inf. Syst.*, 32(2):351–382, 1 Aug. 2012. DOI: 10.1007/s10115-011-0433-1. [cited on page 44]
359. Y.-X. Wang and Y.-J. Zhang. Nonnegative matrix factorization: A comprehensive review. *IEEE Trans. on Knowledge and Data Engineering*, 25(6):1336–1353, Jun. 2013. [cited on page 46]
360. D.J. Watts and S. Strogatz. Collective dynamics of "small-world" networks. *Nature*, 393(6684):440–442, Jun. 1998. DOI: 10.1038/30918. [cited on page 245]
361. W. Wei and J.M. Mendel. Optimality tests for the fuzzy c-means algorithm. *Pattern Recognition*, 27(11):1567–1573, 1994. [cited on page 104, 107]
362. M.P. Windham. Numerical classification of proximity data with assignment measures. *J. Classification*, 2:157–172, 1985. [cited on page 122, 124]

363. I.H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques. Second Edition.* Morgan Kaufmann, 2005. [cited on page 96]
364. F. Wu and B.A. Huberman. Finding communities in linear time: a physics approach. *The European Physical Journal B*, 38(2):331–338, 2004. [cited on page 8, 196, 197]
365. J. Wu, J. Chen, H. Xiong, and M. Xie. External validation measures for k-means clustering: A data distribution perspective. *Expert Systems with Applications*, 36(3):6050–6061, Apr. 2009. [cited on page 132]
366. K.L. Wu and M.S. Yang. Alternative c-means clustering algorithms. *Pattern Recognition*, 35(10):2267–2278, 2002. [cited on page 109, 120]
367. L. Wu, X. Ying, X. Wu, and Z.-H. Zhou. Line orthogonality in adjacency eigenspace with application to community partition. In *Proc. of the 22-nd Intl Joint Conf. on Artificial Intelligence – Vol. 3*, IJCAI'11, pages 2349–2354. AAAI Press, 2011. doi: 10.5591/978-1-57735-516-8/IJCAI11-391. [cited on page 178]
368. X. Wu, V. Kumar, J.R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G.J. McLachlan, A. Ng, B. Liu, P.S. Yu, Z.-H. Zhou, M. Steinbach, D.J. Hand, and D. Steinberg. Top 10 algorithms in data mining. *Knowledge and Information Systems*, 14(1):1–37, 2007. DOI: 10.1007/s10115-007-0114-2. [cited on page 69]
369. Z. Wu and R. Leahy. An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 15(11):1101–1113, Nov. 1993. [cited on page 156]
370. X.L. Xie and G. Beni. A validity measure for fuzzy clustering. *IEEE. Trans. on Pattern Analysis and Mach. Intell.*, 13(8):841–847, 1991. [cited on page 132, 134]
371. E.P. Xing and M.I. Jordan. On semidefinite relaxation for normalized k-cut and connections to spectral clustering. Technical Report UCB/CSD-3-1265, Computer Sci. Division (EECS), University of California, Berkeley, CA 94720, Jun. 2003. [cited on page 45]
372. L. Xu, W. Li, and D. Schuurmans. Fast normalized cut with linear constraints. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR-09)*, pages 2866 – 2873, Miami, FL, USA, 20-25 Jun. 2009. DOI: 10.1109/CVPR.2009.5206561. [cited on page 13, 213, 214]
373. R. Xu and D. Wunsch II. Survey of clustering algorithms. *IEEE Trans. on Neural Networks*, 16(3):645–678, May 2005. [cited on page 21, 25, 59, 114, 129]
374. W. Xu, X. Liu, and Y. Gong. Document clustering based on non-negative matrix factorization. In *Proc. of the 26th Annual Int. ACM SIGIR Conf. on Research and Development in Informaion Retrieval*, pages 267–273, New York, NY, USA, 2003. ACM Press. [cited on page 46]
375. F. Yang, T. Sun, and Ch. Zhang. An efficient hybrid data clustering method based on K-harmonic means and particle swarm optimization. *Expert Systems with Applications*, 36:9847–9852, 2009. [cited on page 87]
376. M.-S. Yang. A survey of fuzzy clustering. *Mathematical and Computer Modelling*, 18(11):1–16, Dec. 1993. DOI: 10.1016/0895-7177(93)90202-A. [cited on page 109]
377. J. Yu, Q. Cheng, and Huang H. Analysis of the weighting exponent in the FCM. *IEEE. Trans. on Syst., Man Cybern. B, Cybern.*, 34(1):634–638, 2004. [cited on page 102]
378. J. Yu and M.S. Yang. Optimality test for generalized FCM and its application to parameter selection. *IEEE. Trans. Fuzzy Syst.*, 13(1):164–176, 2005. [cited on page 104]
379. S.X. Yu and J. Shi. Multiclass spectral clustering. In *Proc. of the IEEE Int. Conf. on Computer Vision*, volume 1, pages 313–319, Nice, France, 11-17 Oct. 2003. [cited on page 146, 173]

380. W.W. Zachary. An information flow model for conflict and fission in small groups. *J. of Anthropological Research*, 33:452–473, 1977. [cited on page 209]
381. R. Zass and A. Shashua. A unifying approach to hard and probabilistic clustering. In *Proc. 10th IEEE Int. Conf. on Computer Vision*, volume 1 of *ICCV 2005*, pages 294–301, Beijing, China, 17-21 Oct. 2005. [cited on page 44, 45]
382. L. Zelnik-Menor and P. Perona. Self-tuning spectral clustering. In *Advances in Neural Information Processing Systems 17*, pages 1601–1608. MIT Press, 2000. [cited on page 188]
383. B. Zhang. Generalized k-harmonic means – boosting in unsupervised learning. Technical Report HPL-2000-137, Hewlett-Packard Labs, 2000. URL: `http://www.hpl.hp.com/techreports/2000/HPL-2000-137.html`. [cited on page 47, 85, 86, 87]
384. B. Zhang. Dependence of clustering algorithm performance on clusteredness of data. Technical Report HPL-2001-91, Hewlett-Packard Labs, 2001. [cited on page 66, 67]
385. D. Zhang and S. Chen. Clustering incomplete data using kernel-based fuzzy c-means algorithm. *Neural Processing Letters*, 18(3):155–162, Dec. 2003. DOI: 10.1023/B:NEPL.0000011135.19145.1b. [cited on page 117, 118]
386. R. Zhang and A.I. Rudnicky. A large scale clustering scheme for kernel k-means. In *Proc. of the 16th Int. Conf. on Pattern Recognition*, volume 4 of *ICPR'02*, pages 40289–40292, Washington, DC, USA, 11-15 Aug. 2002. IEEE Computer Society. [cited on page 88]
387. L. Zhao, H. Nagamochi, and T. Ibaraki. Greedy splitting algorithms for approximating multiway partition problems. *Math. Programming*, 102(1):167–183, 2005. doi: 10.1007/s10107-004-0510-2. [cited on page 171]
388. Q. Zhao. *Cluster validity in clustering methods.* PhD thesis, University of Eastern Finland, Jun. 2012. URL: `http://epublications.uef.fi/pub/urn_isbn_978-952-61-0841-4/urn_isbn_978-952-61-0841-4.pdf`. [cited on page 98, 132]
389. W. Zhao, H. Ma, and Q. He. Parallel k-means clustering based on MapReduce. In M.G. Jaatun, G. Zhao, and C. Rong, editors, *Proc. of the 1st Int. Conf. on Cloud Computing*, volume 5931 of *LNCS*, pages 674–679. Springer-Verlag Berlin, Heidelberg, 2009. DOI: 10.1007/978-3-642-10665-1_71. [cited on page 80]
390. S. Zhong. Efficient online spherical k-means clustering. In *Proc. IEEE Int. Joint Conf. Neural Networks*, IJCNN 2005, pages 3180–3185, Montreal, Canada, 31 Jul. - 4 Aug. 2004. IEEE Computer Society. [cited on page 85]
391. D. Zhou, J. Huang, and B. Schölkopf. Learning from labeled and unlabeled data on a directed graph. In *Proc. of the 22nd Int. Conf. on Machine Learning*, ICML '05, pages 1036–1043, Bonn, Germany, 7-11 Aug. 2005. ACM, New York. DOI: 10.1145/1102351.1102482. [cited on page 13, 212, 213, 251, 259]
392. D. Zhou and B. Schölkopf. Learning from labeled and unlabeled data using random walks. In C.E. Rasmussen, H.H. Bülthoff, B. Schölkopf, and M.A. Giese, editors, *Proc. 26th DAGM Symposium on Pattern Recognition, Tübingen, Germany*, volume 3175 of *LNCS*, pages 237–244, Bonn, Germany, 30 Aug. - 1 Sep. 2004. Springer. DOI: 10.1007/978-3-540-28649-3_29. [cited on page 13, 211, 212, 255]
393. D. Zhou, J. Weston, A. Gretton, O. Bousquet, and B. Schöolkopf. Ranking on data manifolds. In L. Saul, S. Thrun, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, pages 169–176. MIT Press, Cambridge, Mass., 2004. [cited on page 213, 214]
394. S. Zhou and J.Q. Gan. Mercer kernel, fuzzy c-means algorithm, and prototypes of clusters. In Z.R. Yang, H. Yin, and R.M. Everson, editors, *Intelligent Data*

Engineering and Automated Learning - IDEAL 2004, volume 3177 of *LNCS*, pages 613–618. Springer, 2004. [cited on page 119]

395. X. Zhou, M. Belkin, and N. Srebro. An iterated graph laplacian approach for ranking on manifolds. In *Proc. of the 17th ACM SIGKDD Intnl Conf. on Knowledge Discovery and Data Mining*, KDD-11, pages 877–885, San Diego, California, USA, 12-24 Aug. 2011. ACM New York, NY, USA. [cited on page 214]

Index

3

MONOGRAPH SERIES: S.T. WIERZCHOŃ, M.A. KŁOPOTEK ALGORITHMS OF CLUSTER ANALYSIS

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

PhD STUDIES

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

The Project is co-financed by the European Union from resources of the European Social Found

ISBN 978-83-63159-10-8
e-ISBN 978-83-63159-11-5