

Oleksandr Petrov  
Oleksandr Shumeyko  
Beata Basiura  
Anton Petrov

**INTRODUCTION  
TO DATA MINING**

*Oleksandr Petrov, Oleksandr Shumeyko, Beata Basiura, Anton Petrov*

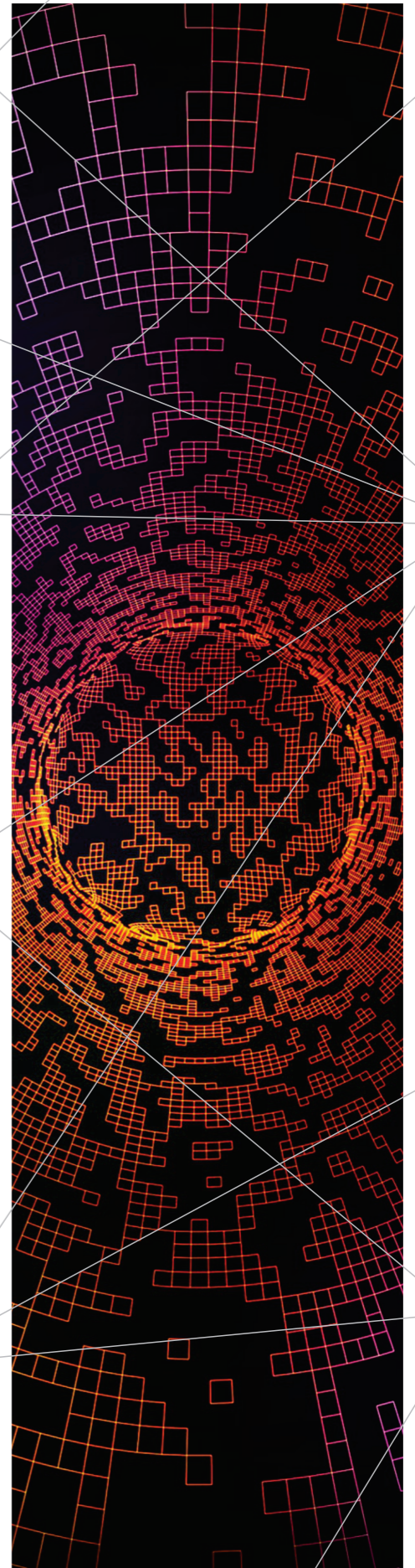
# **INTRODUCTION TO DATA MINING**

ISBN 978-83-66364-25-7



9 788366 136425 7

 WYDAWNICTWA AGH  
KRAKÓW 2019



**Oleksandr Petrov**  
**Oleksandr Shumeyko**  
**Beata Basiura**  
**Anton Petrov**

# **INTRODUCTION** **TO DATA MINING**



WYDAWNICTWA AGH  
KRAKOW 2019



Published by AGH University of Science and Technology Press

© Wydawnictwa AGH, Kraków 2019

ISBN 978-83-66364-25-7

Editor-in-Chief: *Jan Sas*

Editorial Committee:

*Andrzej Pach* (Chairman)

*Jan Chłopek*

*Barbara Gąciarz*

*Bogdan Sapiński*

*Stanisław Stryczek*

*Tadeusz Telejko*

Reviewers:

*prof. dr hab. inż. Mikołaj Karpiński*

*dr hab. inż. Grzegorz Ginda*

Authors' affiliations:

*Oleksandr Petrov – AGH University of Science and Technology in Krakow*

*Oleksandr Shumeyko – Dniprovsk State Technical University*

*Beata Basiura – AGH University of Science and Technology in Krakow*

*Anton Petrov – Federal State Budgetary Educational Institution of Higher Education „Kuban State Agrarian University named after I.T. Trubilin”*

Technical editor: *Kamila Zimnicka*

Desktop Publishing: *Andre*

Cover Design: *Paweł Sepielak*

---

AGH University of Science and Technology Press (Wydawnictwa AGH)

al. A. Mickiewicza 30, 30-059 Kraków

tel. 12 617 32 28, 12 636 40 38

e-mail: [redakcja@wydawnictwoagh.pl](mailto:redakcja@wydawnictwoagh.pl)

<http://www.wydawnictwo.agh.edu.pl>

---

# Contents

<b>Introduction</b> .....	7
<b>1. Least-squares method</b> .....	11
1.1. Ordinary least square method .....	11
1.2. Linearization at the least squares method .....	17
1.3. Examples in Python .....	23
<b>2. Principle Component Analysis</b> .....	32
2.1. The main idea of PCA .....	32
2.2. An iteration scheme of PCA calculating .....	40
2.3. Examples in Python .....	44
2.4. Optimum transition from the RGB model to optimum three-component model .....	48
2.5. Fisher linear discriminant analysis .....	51
2.6. Multidimensional discriminant analysis (MDA) .....	57
<b>3. Application of fuzzy logic in Data Mining</b> .....	72
3.1. What is fuzzy thinking? .....	72
3.2. Fuzzy sets .....	73
3.3. Linguistic variables and linguistic gain .....	77
3.4. Operations on fuzzy sets .....	80
3.5. Properties of operations on fuzzy sets .....	83
3.6. Fuzzy inference rules .....	86
3.7. Defuzzification .....	96
3.8. The choice of alternatives using fuzzy inference rules .....	100
3.9. Ranking alternatives based on heuristic approach .....	110
3.10. Fuzzy decision trees .....	117

<b>4. Soft computing in data handling</b> .....	129
4.1. Introduction to soft computing .....	129
4.2. Evolutionary calculations .....	131
4.2.1. General Introduction .....	131
4.2.2. Genetic algorithm .....	133
4.2.3. Simple example of implementation of GA .....	136
4.2.4. Closer to reality, or the space crossover .....	140
4.2.5. Genetic programming .....	149
4.2.6. To be, or not to be... ..	152
4.2.7. Diophantine equation .....	159
4.3. Swarm intelligence .....	162
4.3.1. The use of ant algorithm for the Traveling Salesman Problem .....	167
<b>5. Clustering methods</b> .....	174
5.1. Clustering. General concepts .....	175
5.2. Hierarchical methods .....	178
5.2.1. Hierarchical methods. Agglomerative algorithms .....	178
5.2.2. Hierarchical methods. Divisive algorithms .....	179
5.3. Examples in Python – clustering hierarchical methods .....	180
5.4. Nonhierarchical algorithms .....	216
5.4.1. <i>K</i> -means method .....	217
5.4.2. Fuzzy <i>k</i> -means .....	220
5.4.3. Gyustafsona–Kessel’s clustering .....	221
5.4.4. Method of correlation galaxies .....	223
5.4.5. Spectral clustering method .....	224
5.5. Examples in Python – clustering nonhierarchical methods .....	228
<b>6. Classifiers</b> .....	248
6.1. Definition of the classification problem .....	248
6.2. Main directions of the research of the classification issue .....	249
6.3. Stochastic classifiers .....	251
6.3.1. Use of the theorem of Bayes for decision-making .....	251
6.4. Naive Bayesian classifier .....	255
6.4.1. Example of sale of the Naive Bayes classifier .....	258
6.4.2. EM algorithm .....	263
6.5. Linear discriminant analysis .....	271
6.5.1. Example 5.1 .....	274
6.5.2. Example 5.2 .....	276
6.5.3. Example 5.3 .....	277

<b>7. Use of genetic algorithms for creation of the vector classifiers .....</b>	<b>279</b>
7.1. Use of Veronoi polyhedron	
in the problem of texts classification .....	282
7.2. Check of the existing classification on the correctness .....	284
<b>8. Support vector machines .....</b>	<b>287</b>
8.1. The main idea .....	287
8.2. SVM for linear separable set .....	289
8.3. SVM for nonlinear separable set .....	290
8.4. Example .....	294
<b>9. Visualization of multidimensional data .....</b>	<b>297</b>
9.1. Multidimensional scaling technic .....	297
9.2. Kohonen self-organizing maps (SOM).....	299
9.2.1. Initialization of the map of Kohonen .....	301
9.2.2. The training algorithm .....	302
9.3. Examples .....	305
9.3.1. Showing similarity of objects .....	305
9.3.2. Showing similarity of European countries .....	306
9.3.3. The world map of poverty .....	308
9.3.4. The traveling salesman problem .....	309
<b>10. Recommender systems .....</b>	<b>311</b>
10.1. General structure of recommendation system .....	313
10.1.1. Collaborative filtering .....	314
10.1.2. The content-oriented recommendations .....	321
10.1.3. Profiles of users .....	322
10.1.4. Training a user model .....	323
10.1.5. Hybrid approaches .....	328
10.2. Analysis of client environments .....	330
10.2.1. Examples of client environments .....	330
10.2.2. Retail chain stores .....	331
10.2.3. Mobile operators .....	331
10.2.4. Online stores of books, audio and video of other products .....	331
10.2.5. Search engines .....	332
10.2.6. Parliamentary elections .....	332
10.2.7. Analysis of texts .....	333
10.2.8. Social networks .....	333



<b>Appendix</b>	
<b>Basic information</b> .....	334
A.1. Background information on linear algebra .....	334
A.2. Background information on probability theory .....	336
<b>References</b> .....	347
<b>CD contents</b> .....	364

## Introduction

*Πάντα ῥεῖ καὶ οὐδέν μένει* – everything flows, nothing stands still, refers to Heraclitus' approach highlighted in his meaningful quote describing most profoundly the events of existing world. In the 21 century, our world is undergoing major changes, although the most radical and large-scale transformations, first and foremost, deal with the field of information society.

Back in the 1920 of the last century, Academician VI Vernadsky pointed out that there is a strong human impact on the environment as well as the transformation of the modern biosphere. One of his conclusions was the postulate that says mankind as a biosphere element inevitably covers the reasonable control of the planet's living skin, turning it into one sphere – the no sphere (the sphere of reason). Those changes that we are able to notice in the field of information, truly confirm the Vernadsky's conclusions. The significant shifts in information society (and especially within information technology industries) are not just another move towards the development of scientific and technological revolution, but also have a global civilizational nature. It is anticipated that by the middle of the 21st century, our planet will create a completely new kind of civilization – an information civilization. The formation process of the civilization will highly increase the significance of information and scientific knowledge in almost all spheres of society.

So what information is about? In a narrower sense, information is defined as any set of signals impacts or data which a system received from the environment (input information X), issued in the environment (Y output information) and stored within itself (internal, internal system information Z).

In broad terms, the information should be considered as a special kind of resource, as the stock of knowledge including certain material objects or energy, structural or any other object characteristics. Compared to the resources associated with material objects, information resources are inexhaustible requiring substantially different methods of reproduction and renewal than material resources.

The highest form of information is knowledge. The interdisciplinary concept which claims to be included in the most important philosophical category. From philosophical

standpoint knowledge management is primarily seen as one of the functional aspects of management areas.

One of the tools to work through information is Data Mining. The term Data Mining was introduced first by Gregory Piatetskim-Shapiro in 1989. It is primarily used to designate a plurality of information and feature extraction methods from a large number of poorly structured data source. Specifically Data Mining methods are quite easily fragmented including systems: Web Mining, Text Mining etc.

The core of Data Mining techniques includes all sorts of classification methods, modeling and prediction, based on the use of decision trees, artificial neural networks, genetic algorithms, evolutionary programming, associative memory, fuzzy logic. The Data Mining techniques often use statistical methods (descriptive analysis, correlation and regression analysis, factor analysis, variance analysis, component analysis, discriminant analysis, time series analysis, survival analysis, communications analysis). Such methods, however, require some notions of a priori for the analyzed data, which is somewhat at odds with the objectives of the Data Mining (detection of previously unknown non-trivial and practically useful knowledge).

One of the most essential postulates of Data Mining methods is a visual representation of the calculations results (visualization) that allows people, who have no special mathematical training, use the Data Mining toolkit. At the same time, the use of statistical methods referring to data analysis calls for applying probability and mathematical statistics.

Data mining techniques can be used for working with large data, and through relatively small amounts of data (obtained, e.g., by results of separate experiments or the analysis about the company data). A criterion for a sufficient amount of data is labeled as a field of study and analysis algorithm which is being applied these days.

The methods of Data Mining, are primarily implemented to solve problems which are divided into either descriptive or predictive ones.

The descriptive tasks, most importantly, give a clear explanation of the existing hidden patterns, while the nature of predictive tasks is predominantly based on a question about the prediction of the cases for which information was not delivered yet.

Descriptive tasks include:

- search association rules or patterns (samples);
- grouping objects, cluster analysis;
- construction of a regression model.

Predictive tasks include:

- classification of objects (for predetermined classes);
- regression analysis, time series analysis.

The monograph is a basic information about a method of least squares, principal component analysis, linear discriminant Fischer analysis, the use of fuzzy logic in Data Mining, soft calculation in data processing, clustering methods, classifiers, decision trees, support vector machines, visualization of multidimensional data and recommending system.

## **Preface**

Basically, the monography gives an insight into the research dedicated to the fields of Data Mining included within 10 sections. The appending is designed as an introduction giving an overview of elementary maths applied for data transformation method. For better comprehending of the book, we outlined the basis of lineal algebra as well as probability calculation. As for the first section, it introduces the Least Squared Method either in elementary model or estimated model type. In particular, it discusses model using the type of spline piecewise line or polygonal linear functions.

In the face of collecting data into big data sets, Factor Analysis and Principal Component Analysis are presented in the second section as solutions for reducing data size. The unit demonstrates the example based on the transition of RGB model to three-component model. The chapter discusses examples of the Fisher's Linear Discriminant method used for compressing and recognizing image.

These days, artificial intelligence has been widely seen as one of the most essential tasks. Presently, the challenge for programmer is based on designing such a system which is able to consider human subjective valuation, for instance, referring to weather or product reviews. For example, implications, such as: „If there is a nice weather, then I will go for playing tennis but If there is a bad weather, then I will go to the swimming pool”, include the uncertainty of the choice whether I should go to the swimming pool or not. Such linguistic variable and linguistic gains can be implemented by means of fuzzy logic, which is able to distinguish the uncertainty of human choices.

The third section consists of examples referring to operations based on fuzzy set as well as fuzzy rules. Not to mention that Mamdani, Tsukamoto, Sugeno and Larsen's systems were used for selecting financing stylesheets. Besides, the unit is extended to alternative heuristic rank method. Also, the part describes fuzzy decision tree used for selecting applicants with proper qualifications and salary expectations.

The next section introduces the subjects of general Soft Computing including evaluation, genetics and ants algorithms as well as neural networks. The cornerstone of this part is based on the examples given in Python, which are available to be downloaded CD.

The 5th part includes clustering methods beginning with setting the object similarity measure, through different clustering methods such as hierarchical methods,



optimization methods (k-means and fuzzy k-means) to proceeding to maximizing correlation communication. Considered example is based on clustering of images.

The next section gives an overview of classification problems such as Bayes native classifier and its extension to EM algorithm. The Linear Discriminant methods are used and extended by adding nonlinear models. The linear approach was applied for distinguishing reviews left on websites into negative and positive ones. This unit discusses three implemented examples in Python.

An application of genetics algorithms in classification methods is demonstrated in 7th section. As the example, the section shows text classification. It is extended with polyhedron Black area used for solving the problem referring to classification of texts. Moreover, the comparison using the base of Reuters documents is demonstrated there.

Support Vector Machine is the issue presented in 8th section. The chapter shows applying this method for linear separable and non-linear separable sets.

These days, the major task is to approximate the interpretation and visualization of the estimated conclusions. In large data sets, the interpreting and presenting the obtained results has been a great challenge. Plotting the results on a plane seems to be a solution, especially when it comes to retaining the objects similarity. Section 9th gives a good grounding for examples of multidimensional scaling applying Kohonen's self-organizing maps on the example of World map of poverty. This is one of the options that allows to find a solution for traveler salesman problem.

The last chapter focusses on modelling and applying recommendation systems. In the light of collecting all types of data, i.e., tracking websites, verifying social media inquiries, searching for hidden preference profiles, such systems are a great importance. The definition of user profiles is shown on the example using the known methods of data mining for selecting mobile network operators, selecting videos, or for choosing the audiobook to be listened to. Besides, selected ranking systems are presented on the analysis of social media text.

As examples of the implementation of the algorithms presented in the book there are given source code of programs in Python.

# 1. Least-squares method

## 1.1. Ordinary least square method

Let the following system of points be given (see Tab. 1.1) where the number of points  $N$  is big and data are obtained with the mistake.

**Table 1.1**  
Experimental data

$x_i$	$x_0$	$x_1$	...	$x_N$
$t_i$	$t_0$	$t_1$	...	$t_N$

Such situation is common when processing experimental data. For that matter, the use of interpolation methods is inexpedient. We should add to this, the situation when the priori information in the studied process is accessible and the nature of the target device is defined by technological conditions or the nature of the phenomenon. The choice of the estimated function coefficients is determined, first of all, by adequacy to the used model, that is, the error between the input data and its description has to be relatively small. Clearly, the choice of proximity criterion is the most essential. As a rule, the solution for such tasks relies on mean square distance. It is caused, first of all, by the fact that in this case, when using linear methods of approach, function of the purpose is represented by square function – the paraboloid. Owing to camber, the paraboloid has the only extremum therefore the necessary condition of the extremum matches with the sufficient one that significantly simplifies the problem of search of the minimum value of objective function. Respectively, the method of finding the extremum for this objective function is called the least-squares method (see in Björck 1996, *Encyclopedia Britannica* 2018, Lawson 1987, Miller 2006, *Ordinary least...*, Robinson 1983, Wolberg 2006).

Now, we turn to the method of least squares. The purpose of the method is to determine a function of form below (see eq. (1.1)) in accordance with the initial data given in Table 1.1.

$$F(\{a_i\}_{i=0}^n, t) = \sum_{i=0}^n a_i \phi_i(t) = a_0 \phi_0(t) + a_1 \phi_1(t) + \dots + a_n \phi_n(t) \quad (1.1)$$

where:

$\phi_i(t), i = 0, \dots, n$  – are the basic functions,  
 $a_i$  – the unknown coefficients which are subject to definition.

In particular, if we use degree monomials as basic functions  $\phi_i(t) = t^i$ , the task comes down to search for the polynomial with degree  $n$  in following form

$$F(\{a_i\}_{i=0}^n, t) = \sum_{i=0}^n a_i t^i = a_0 + a_1 t + \dots + a_n t^n,$$

which estimates points from the initial table.

For finding coefficients  $a_i$  let's look at the function  $F(\{a_i\}_{i=0}^n, t)$  (shown in Fig. 1.1), for which the deviation between the values of function and the values set in the Table 1.1 is the smallest in integral sense.

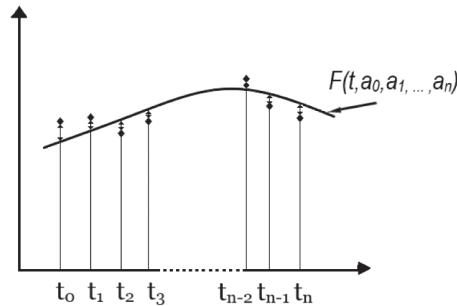


Fig. 1.1. Illustration of LSM

In particular, in the discrete least-squares method the objective function is under the following construction

$$\begin{aligned} S(a_0, a_1, \dots, a_n) &= \sum_{i=0}^N (F(a_0, a_1, \dots, a_n, t_i) - x_i)^2 \rho_i^2 = \\ &= \sum_{i=0}^N (a_0 \phi_0(t_i) + a_1 \phi_1(t_i) + \dots + a_n \phi_n(t_i) - x_i)^2 \rho_i^2 \end{aligned} \quad (1.2)$$

where  $\rho_i$  – some non-negative numbers (weighting coefficients).

If all deviation are equal zero, then weighting coefficients are equal to one.

Geometrically the objective function (1.2) represents the sum of squares of deviations between experimental data  $x_i$  and values of the approximating function  $F(a_0, a_1, \dots, a_n, t)$  in points  $t_i$  ( $i = 0, \dots, N$ ) with the weight  $\rho_i$ .

For finding the minimum of multivariable, let's look at the function

$$S(a_0, a_1, \dots, a_n) \rightarrow \min_{a_0, a_1, \dots, a_n},$$

the first-order partial derivative of objective function  $S(a_0, a_1, \dots, a_n)$  with respect to each  $a_i$  are equal zero i.e.:

$$\left\{ \begin{array}{l} \frac{\partial S}{\partial a_0} = 2 \sum_{i=0}^N (F(a_0, a_1, \dots, a_n, t_i) - x_i) \phi_0(t_i) \rho_i^2 = 0, \\ \frac{\partial S}{\partial a_1} = 2 \sum_{i=0}^N (F(a_0, a_1, \dots, a_n, t_i) - x_i) \phi_1(t_i) \rho_i^2 = 0, \\ \dots\dots\dots \\ \frac{\partial S}{\partial a_n} = 2 \sum_{i=0}^N (F(a_0, a_1, \dots, a_n, t_i) - x_i) \phi_n(t_i) \rho_i^2 = 0. \end{array} \right. \quad (1.3)$$

These system of equations (1.3) is a necessary condition (and, in this case, owing to convexity of the objective function, also sufficient condition) for determining this minimum.

The received system of equations represents the system of linear algebraic equations with  $n + 1$  unknowns  $a_0, a_1, \dots, a_n$ . The system is solvable under the condition  $n \leq N$ . Its matrix is symmetric and positive definite. Parameters  $a_0, a_1, \dots, a_n$  ensure minimization of equation (1.2).

The solution of this system can be carried out by any of the known methods (for example, Gauss's method, Cramer's rule, etc.). Substituting in (1.1)  $a_0, a_1, \dots, a_n$  with values found as the solution of linear system (1.3), we receive the  $F(t)$  function that is the best approximation of the input data from Table 1.1 in mean square sense. Quality of such approach can be evaluated by the size of root-mean-square deviation

$$\sigma = \sqrt{\frac{1}{N+1} \sum_{i=0}^N (x_i - F(t_i))^2 \rho_i^2}.$$

Quite often, natural assumption is that the input data is the most efficiently modeled by a straight line or a parabola. In this case, we say that linear or square regression is used.



Let's consider the case of the priori data description for the straight line used by the method of linear regression. Let's the input data  $(t_i, x_i), i = 0, 1, \dots, N$  be approximated by straight line  $x = at + b$ . In this case the objective function (1.2) will take the form

$$S(a, b) = \sum_{i=0}^N (at_i + b - x_i)^2 \rightarrow \min_{a, b}.$$

The necessary (and, in this case, sufficient) condition of the extremum can be written as follows:

$$\begin{cases} \frac{\partial}{\partial a} S(a, b) = 2 \sum_{i=0}^N t_i (at_i + b - x_i) = 0, \\ \frac{\partial}{\partial b} S(a, b) = 2 \sum_{i=0}^N (at_i + b - x_i) = 0, \end{cases}$$

or that the same:

$$\begin{cases} a \sum_{i=0}^N t_i^2 + b \sum_{i=0}^N t_i = \sum_{i=0}^N x_i t_i, \\ a \sum_{i=0}^N t_i + b(N+1) = \sum_{i=0}^N x_i. \end{cases}$$

As the result, applying Cramer's rule in solving of linear equations system, we receive straight-line coefficients (linear regression):

$$a = \frac{(N+1) \sum_{i=0}^N x_i t_i - \sum_{i=0}^N t_i \sum_{i=0}^N x_i}{(N+1) \sum_{i=0}^N t_i^2 - \left( \sum_{i=0}^N t_i \right)^2},$$

$$b = \frac{\sum_{i=0}^N t_i^2 \sum_{i=0}^N x_i - \sum_{i=0}^N t_i \sum_{i=0}^N x_i t_i}{(N+1) \sum_{i=0}^N t_i^2 - \left( \sum_{i=0}^N t_i \right)^2}.$$

Generally, widespread approach is based on piecewise polynomial functions or splines. The most widespread type of splines are piecewise or polygonal linear functions. As regression model, let's consider the broken line with constant *weigh function*

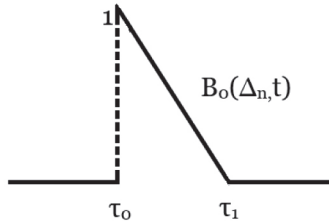
equal to one. Let  $\Delta_n$  be the fixed splitting the interval  $[t_0, T]$  at points  $\tau_i$  ( $i = 0, 1, 2, \dots, n$ ), and  $\mathfrak{R}(\Delta_n)$  set of broken lines  $P(\Delta_n, t) = P(\{a_i\}_{i=0}^n, \Delta_n, t)$  with nodes in splitting points  $\Delta_n$ . Then the problem related to finding piecewise linear regression model with the fixed nodes looks as follows

$$\inf \left\{ \sum_{i=0}^N (x_i - P(\Delta_n, t_i))^2 \mid P(\Delta_n) \in \mathfrak{R}(\Delta_n) \right\}.$$

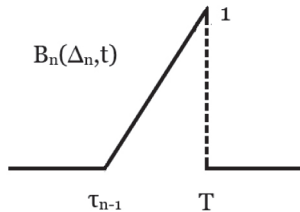
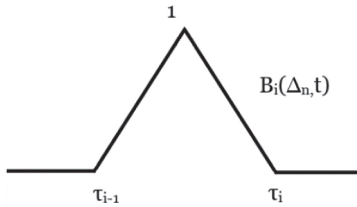
It is easy to notice that

$$P(\Delta_n, t) = P(\{a_i\}_{i=0}^n, \Delta_n, t) = \sum_{i=0}^n a_i B_i(\Delta_n, t),$$

where  $B_i(\Delta_n, t)$  ( $i = 0, \dots, n$ ) are basic functions which can be written down in the following way:



$$B_0(\Delta_n, t) = \begin{cases} (\tau_1 - t)(\tau_1 - \tau_0)^{-1}, & t \in [\tau_0, \tau_1], \\ 0, & \text{otherwise,} \end{cases}$$



$$B_n(\Delta_n, t) = \begin{cases} (\tau_{n-1} - t)(\tau_{n-1} - T)^{-1}, & t \in [\tau_{n-1}, T]. \\ 0, & \text{otherwise,} \end{cases}$$

Let's write the objective function

$$S(a_0, a_1, \dots, a_n) = \sum_{i=0}^N \left( \sum_{j=0}^n a_j B_j(\Delta_n, t_i) - x_i \right)^2,$$

also we will find the solution of the task  $S(a_0, a_1, \dots, a_n) \rightarrow \min_{a_0, a_1, \dots, a_n}$ . Necessary and sufficient condition of the extremum looks as follows

$$\frac{\partial}{\partial a_i} S(a_0, a_1, \dots, a_n) = 0, \quad i = 0, 1, \dots, n.$$

Finding of the extremum comes down to the solution of combined equations

$$\begin{pmatrix} \langle B_0, B_0 \rangle & \langle B_0, B_1 \rangle & \cdots & \langle B_0, B_n \rangle \\ \langle B_1, B_0 \rangle & \langle B_1, B_1 \rangle & \cdots & \langle B_1, B_n \rangle \\ \vdots & \vdots & \ddots & \vdots \\ \langle B_n, B_0 \rangle & \langle B_n, B_1 \rangle & \cdots & \langle B_n, B_n \rangle \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} \langle x, B_0 \rangle \\ \langle x, B_1 \rangle \\ \vdots \\ \langle x, B_n \rangle \end{pmatrix},$$

where

$$\langle x, B_j \rangle = \sum_{i=0}^N x_i B_j(\Delta_n, t_i), \quad j = 0, 1, \dots, n,$$

and noticing that  $\langle B_i, B_j \rangle = 0, \forall i, j: |i - j| \geq 2$  we get combined equations with the three-scalar matrix

$$A = \begin{pmatrix} \langle B_0, B_0 \rangle & \langle B_0, B_1 \rangle & 0 & \cdots & 0 \\ \langle B_1, B_0 \rangle & \langle B_1, B_1 \rangle & \langle B_1, B_2 \rangle & \cdots & 0 \\ 0 & \langle B_2, B_1 \rangle & \langle B_2, B_2 \rangle & \cdots & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & 0 & \cdots & \langle B_n, B_n \rangle \end{pmatrix}.$$

After applying the sweep method, we receive the effective algorithm finding the equation of piecewise linear regression with the fixed nodes.

Let's give the example of creating the broken line with the least-squares method (see Fig. 1.2).

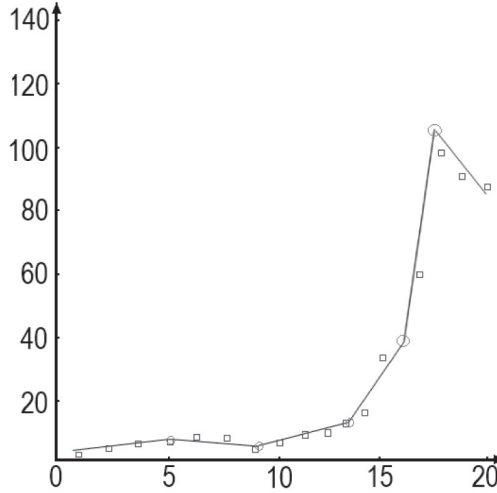


Fig. 1.2. Application of discrete data for the broken line on LSM

## 1.2. Linearization at the least squares method

The given above methodology referring to the approximating functions by the least squares method suits only for functions at which undetermined coefficients are set linearly. If this condition is not satisfied, then direct use of the least squares method is impossible.

How should it look like in this case? Whether in general it is possible to use the least-squares method? Presumably yes. But unfortunately, it is necessary to apply some additional constructions linearizing (on coefficients) the approximating function (see Hastie et al. 2009, Rao, Toutenburg 1999, Lawson 1987, Wolberg 2006).

Several examples are given below.

Let's also assume that the function is written as follows  $x = 1/(\alpha t + \beta)$ .

Then for  $x_i$  the error will be calculated by using the following formula

$$\delta_i = x_i - \frac{1}{\alpha t_i + \beta} \quad (1.4)$$

Direct use of the least-squares method leads to minimization of the following expression

$$S(\alpha, \beta) = \sum_{i=1}^n \delta_i^2 = \sum_{i=1}^n \left( x_i - \frac{1}{\alpha t_i + \beta} \right)^2 \quad (1.5)$$



Let's calculate the first order partial derivative of above mentioned function with regards to coefficients  $\alpha$  and  $\beta$ , where these partial derivative are equated to zero, we will receive system of two nonlinear equations:

$$\begin{cases} \frac{\partial S(\alpha, \beta)}{\partial \alpha} = 2 \sum_{i=1}^n \left( x_i - \frac{1}{\alpha t_i + \beta} \right) \frac{t_i}{(\alpha t_i + \beta)^2} = 0, \\ \frac{\partial S(\alpha, \beta)}{\partial \beta} = 2 \sum_{i=1}^n \left( x_i - \frac{1}{\alpha t_i + \beta} \right) \frac{1}{(\alpha t_i + \beta)^2} = 0, \end{cases}$$

which is not subject to the exact decision. In this instance, we will carry out some transformations.

Let's consider values

$$\Delta_i = x_i (\alpha t_i + \beta) - 1, \quad (i = 1, 2, \dots, n).$$

Let's establish dependency between  $\Delta_i$  and  $\delta_i$ . From (1.4) we get

$$\alpha t_i + \beta = \frac{1}{x_i - \delta_i}.$$

Then

$$\Delta_i = \frac{x_i}{x_i - \delta_i} - 1 = \frac{\delta_i}{x_i - \delta_i}, \quad (i = 1, 2, \dots, n)$$

and, therefore, at small  $\Delta_i$

$$\delta_i = \frac{x_i \Delta_i}{\Delta_i + 1} \approx x_i \Delta_i.$$

Then the task (1.5) comes down to the problem related to determining coefficients  $\alpha$  and  $\beta$  so that the objective function

$$\sum_{i=1}^n (x_i \Delta_i)^2 = \sum_{i=1}^n (1 - \alpha t_i x_i - \beta x_i)^2 x_i^2$$

obtain minimum.

Thus, we came to the task (1.2) provided that the approximated function is identically equal to unit and  $\phi_0(t) = \alpha t$ ,  $\phi_1(t) = x(t)$  with the weight  $\rho_i = x_i$ .

The error of the objective function takes the following form

$$\sigma = \sqrt{\frac{1}{n} \sum_{i=1}^n \left( x_i - \frac{1}{\alpha t_i + \beta} \right)^2}.$$

Let's review other example. Let the approximating function be written as

$$x = \frac{t}{\alpha t + \beta}.$$

For  $x_i$  the error should be equal

$$\delta_i = x_i - \frac{t_i}{\alpha t_i + \beta} \tag{1.6}$$

and

$$\Delta_i = x_i(\alpha t_i + \beta) - t_i, \quad (i = 1, 2, \dots, n).$$

Let's establish connection between  $\Delta_i$  and  $\delta_i$ . From (1.6) we have

$$\alpha t_i + \beta = \frac{t_i}{x_i - \delta_i}.$$

Then

$$\Delta_i = \frac{t_i x_i}{x_i - \delta_i} - t_i = \frac{t_i \delta_i}{x_i - \delta_i}, \quad (i = 1, 2, \dots, n).$$

Next, as the result, at small  $\Delta_i$  (a *small* neighborhood of the  $\Delta$ )

$$\delta_i = \frac{x_i \Delta_i}{\Delta_i + t_i} \approx \frac{x_i}{t_i} \Delta_i.$$

For small  $\delta_i$

$$\sum_{i=1}^n \delta_i^2 \approx \sum_{i=1}^n \left( \frac{x_i}{t_i} \Delta_i \right)^2$$

and

$$\sum_{i=1}^n \left( \frac{x_i}{t_i} \Delta_i \right)^2 = \sum_{i=1}^n (t_i - \alpha t_i x_i - \beta x_i)^2 \left( \frac{x_i}{t_i} \right)^2.$$

Thus, we can move on to the task (1.2) provided that the fitted function is identically equal  $t$  and  $\phi_0(t) = tx(t)$ ,  $\phi_1(t) = x(t)$  with  $\rho_i = x_i/t_i$ .

The error takes the form

$$\sigma = \sqrt{\frac{1}{n} \sum_{i=1}^n \left( x_i - \frac{t_i}{\alpha t_i + \beta} \right)^2}.$$

At last, let the approximating function be written as

$$x = \frac{\alpha t + \beta}{\gamma t + 1}.$$

The task involves finding  $\alpha$ ,  $\beta$ ,  $\gamma$  coefficients so that

$$\sum_{i=1}^n \delta_i^2 = \sum_{i=1}^n \left( x_i - \frac{\alpha t_i + \beta}{\gamma t_i + 1} \right)^2$$

will be minimum. We linearize this task.

Let

$$\Delta_i = \gamma t_i x_i + x_i - \alpha t_i - \beta, \quad (i = 1, 2, \dots, n).$$

Let's establish dependence between  $\Delta_i$  and  $\delta_i$ . From previous calculation we receive

$$\frac{\Delta_i}{\gamma t_i + 1} = x_i - \frac{\alpha t_i + \beta}{\gamma t_i + 1}.$$

Thus, we get

$$\frac{\Delta_i}{\gamma t_i + 1} = \delta_i.$$

For small,  $\Delta_i$  we receive the task equivalent to required minimum.

$$\sum_{i=1}^n \left( \frac{\Delta_i}{\gamma_i + 1} \right)^2 = \sum_{i=1}^n (x_i - \alpha t_i - \beta + \gamma_i x_i)^2 \left( \frac{1}{\gamma_i + 1} \right)^2 \rightarrow \min.$$

It is not possible to use the least-squares method as the method of determining the unknown  $\alpha$ ,  $\beta$ ,  $\gamma$  parameters. Therefore, we can consider the iterative method of step-by-step calculation of weighting coefficients.

For the task (1.2) we will put  $\phi_0(t) = t$ ,  $\phi_1(t) = 1$ ,  $\phi_2(t) = -tx(t)$  and  $\rho_i = 1$ .

Solving this problem, we receive first approximation  $\alpha_1$ ,  $\beta_1$ ,  $\gamma_1$ .

Assuming that  $\phi_0(t) = t$ ,  $\phi_1(t) = 1$ ,  $\phi_2(t) = -tx(t)$ ,  $\rho_i = 1/(\gamma_1 t_i + 1)$  and again solving this problem, we receive  $\alpha_2$ ,  $\beta_2$ ,  $\gamma_2$ . Continuing this process at  $\phi_0(t) = t$ ,  $\phi_1(t) = 1$ ,  $\phi_2(t) = -tx(t)$ ,  $\rho_i = 1/(\gamma_2 t_i + 1)$ , we can determine the following values  $\alpha$ ,  $\beta$ ,  $\gamma$ .

We will continue iteration until ratios are carried out:

$$\left\{ \begin{array}{l} |\alpha_k - \alpha_{k-1}| < \varepsilon, \\ |\beta_k - \beta_{k-1}| < \varepsilon, \\ |\gamma_k - \gamma_{k-1}| < \varepsilon, \end{array} \right.$$

where  $\varepsilon$  is the set error.

Naturally, all the set of regression models used is not exhausted by fractional-linear functions, often used of the power and exponential functions.

Let's look for the approximating (estimating) function in the form  $x = \alpha t^\beta$ . For all  $i = 1, 2, \dots, n$  let's put  $\delta_i = x_i - \alpha t_i^\beta$  and

$$\Delta_i = \ln x_i - \ln \alpha - \beta \ln t_i = \ln \frac{x_i}{\alpha t_i^\beta}.$$

As we demonstrated before, we establish connection between these sizes. From the first equality, we get  $\alpha t_i^\beta = x_i - \delta_i$  and substitute them in the second equation

$$\Delta_i = \ln \frac{x_i}{x_i - \delta_i}.$$

Hence,  $x_i - \delta_i = x_i \exp(-\Delta_i)$  at small  $\Delta_i$  we can write down  $\delta_i = x_i (1 - \exp(-\Delta_i)) \approx x_i \Delta_i$ . Thus, problem referring to the minimization of errors  $\sum_{i=1}^n \delta_i^2$  can be replaced with the problem of minimization the following expression

$$\sum_{i=1}^n (x_i \Delta_i)^2 = \sum_{i=1}^n (\ln x_i - \ln \alpha - \beta \ln t_i)^2 x_i^2,$$

which is the task (1.2).

The model error can look as follows

$$\sigma = \sqrt{\frac{1}{n} \sum_{i=1}^n (x_i - \alpha t_i^\beta)^2}.$$

Let the approximation function be a set of monomials in the form  $x = \alpha \beta^t$ . For all  $i = 1, 2, \dots, n$  let's put  $\delta_i = x_i - \alpha \beta^{t_i}$  and  $\Delta_i = \ln x_i - \ln \alpha - t_i \ln \beta = \ln x_i / (\alpha \beta^{t_i})$ .

Let's find the dependence between these errors.

Expressed in the first equality  $\alpha \beta^{t_i} = x_i - \delta_i$  and substituted in the second, we receive

$$\alpha \beta^{t_i} = x_i - \delta_i.$$

Therefore,  $\delta_i = x_i (1 - \exp(-\Delta_i)) \approx x_i \Delta_i$ . Thus, similarly, the problem related to minimization of the sum  $\sum_{i=1}^n \delta_i^2$  can be replaced with the problem minimization of the expression

$$\sum_{i=1}^n (x_i \Delta_i)^2 = \sum_{i=1}^n (\ln x_i - \ln \alpha - t_i \ln \beta)^2 x_i^2,$$

what solves the problem the purpose function (1.2).

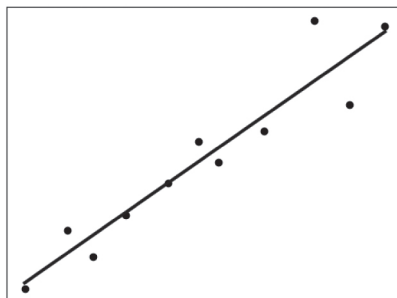
In this case, the error takes the form

$$\sigma = \sqrt{\frac{1}{N} \sum_{i=1}^N (x_i - \alpha \beta^{t_i})^2}.$$

## 1.3. Examples in Python

Several examples given below are inspired by (Pedregosa et al. 2011).

```
#####  
# Example 1.3.1  
# Linear regression (empirical data)  
# company turnover (X) versus transportation cost (y)  
#####  
  
import numpy as np  
from sklearn import linear_model  
import matplotlib.pyplot as plt  
  
X=[[125],[210],[260],[325],[410],[470],[510],[600],[700],[770],[840]]  
y=[30,41,36,44,50,58,54,60,81,65,80]  
  
reg = linear_model.LinearRegression()  
reg.fit(X,y)  
  
# print the parameters  
print(reg.coef_)  
print(reg.intercept_)  
# prin the model  
print('model : y = ', reg.coef_, ' * X + ', reg.intercept_, '\n')  
  
# The mean squared error  
print("Mean squared error: %.2f"  
      % np.mean((reg.predict(X) - y) ** 2))  
# Explained variance score: 1 is perfect prediction  
print('Variance score: %.2f' % reg.score(X, y))  
  
# Plot outputs  
plt.scatter(X, y, color='black')  
plt.plot(X, reg.predict(X), color='blue', linewidth=3)  
plt.show()
```



```
#####
# Example 1.3.2
# Linear regression (generated data)
#####

from sklearn.datasets import make_regression
from sklearn import linear_model
import matplotlib.pyplot as plt
import numpy as np

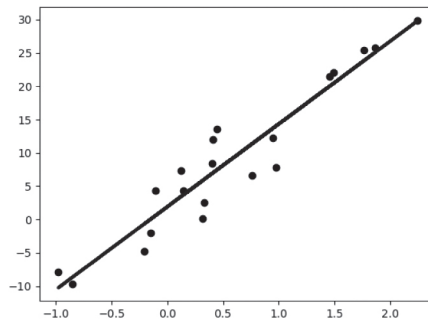
# Generate data
rng = np.random.RandomState(0)
X, y = make_regression(n_samples=20, n_features=1, random_state=0, noise=4.0,
bias=0.0)

reg = linear_model.LinearRegression()
reg.fit(X,y)

# print the parameters:
print(reg.coef_)
print(reg.intercept_)
# print the model
print( 'model : y = ', reg.coef_, ' * X + ', reg.intercept_, '\n')

# The mean squared error
print("Mean squared error: %.2f"
      % np.mean((reg.predict(X) - y) ** 2))
# Explained variance score: 1 is perfect prediction
print('Variance score: %.2f % reg.score(X, y)')

# Plot outputs
plt.scatter(X, y, color='red')
plt.plot(X, reg.predict(X), color='blue', linewidth=3)
plt.show()
```



```

#####
# Example 1.3.3
# polynomial regression (generated data)
#####

import matplotlib.pyplot as plt
import numpy as np
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.pipeline import Pipeline

#define function to determine points with random parameters
def f(x):
    return np.random.uniform(1.75,2.25,size = len(x))*x**3 - np.random.uniform(38,40, size =
        len(x))*x**2 + np.random.uniform(238,242, size = len(x))*x - np.random.normal(0, 1,
        size=len(x))

# generate points used to plot
x_plot = np.linspace(2, 10, 100)

# generate points and keep a subset of them
x = np.linspace(2, 10, 100)
rng = np.random.RandomState(0)
rng.shuffle(x)
x = np.sort(x[:20])
y = f(x)

colors = ['darkorchid', 'mediumvioletred', 'darkcyan']

plt.scatter(x_plot, f(x_plot), color='red',label="ground truth")
plt.scatter(x, y, color='navy', s=30, marker='o', label="training points")

# create matrix versions of the data for regression model
X = x[:, np.newaxis]
X_plot = x_plot[:, np.newaxis]

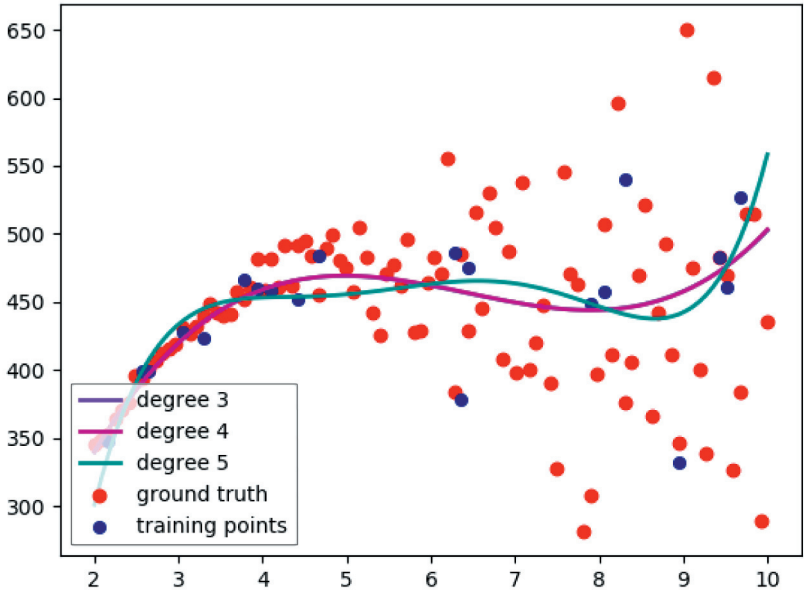
for count, degree in enumerate([ 3, 4, 5]):
    model = Pipeline([('mypoly',PolynomialFeatures(degree)),
        ('mylinear',LinearRegression(fit_intercept=True))])
    model.fit(X, y)
    y_plot = model.predict(X_plot)
    plt.plot(x_plot, y_plot, color=colors[count], linewidth=2, label="degree %d" % degree)

plt.legend(loc='lower left')
plt.show()

```



```
# print the parameters:
print(model.named_steps['mylinear'].coef_)
#fited model: y = 0 - 160x + 77.77x2 - 168.93x3 + 16.89x4 - 0.63x5 + e
```



```
#####
# Example 1.3.4
# polynomial regression (generated data - function cos)
#####
```

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.pipeline import Pipeline

#define function to determine points with random parameters
def r(x):
    return np.random.uniform(0.75,1.25,size = len(x))*x* np.cos(x) + np.random.normal(0, 1,
        size=len(x))

# generate points used to plot
x_plot = np.linspace(2, 10, 100)
```

```

# generate points and keep a subset of them
x = np.linspace(2, 10, 100)
rng = np.random.RandomState(0)
rng.shuffle(x)
x = np.sort(x[:20])
y = r(x)

colors = ['darkorchid', 'mediumvioletred', 'darkcyan']

plt.scatter(x_plot, r(x_plot), color='red',label="ground truth")
plt.scatter(x, y, color='navy', s=30, marker='o', label="training points")

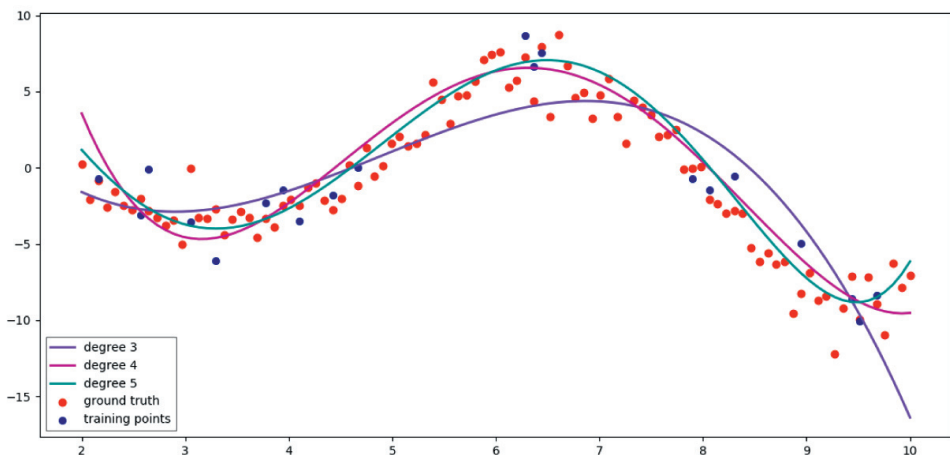
# create matrix versions of the data for regression model
X = x[:, np.newaxis]
X_plot = x_plot[:, np.newaxis]

for count, degree in enumerate([ 3, 4, 5]):
    model = Pipeline([('mypoly',PolynomialFeatures(degree)),
                     ('mylinear',LinearRegression(fit_intercept=True))])
    model.fit(X, y)
    y_plot = model.predict(X_plot)
    plt.plot(x_plot, y_plot, color=colors[count], linewidth=2, label="degree %d" % degree)

plt.legend(loc='lower left')
plt.show()

# print the parameters:
print(model.named_steps['mylinear'].coef_)
# fitted model: fitted model: y = 0 + 102.77x - 50.34x^2 + 112.253x^3 - 1.13x^4 - 0.04x^5 + e

```



```

#####
# Example 1.3.5
# nonlinear regression ver 1
#####

import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn import linear_model

def h(x):
    return np.random.uniform(0.5,1.5,size = len(x))/(2*x + 2 ) - 0.005*np.random.normal(0, 1,
        size=len(x))

# generate data
x_plot = np.linspace(2, 10, 100)

# generate points and keep a subset of them
x = np.linspace(2, 10, 100)
rng = np.random.RandomState(0)
rng.shuffle(x)
x = np.sort(x[:20])
y = h(x)

# the linear regression was used
x1 = y
x2 = x * y
y1 = np.ones(20)
w = x*x

dane = pd.DataFrame({'x2' : np.array(x2),
    'x1': np.array(x1), 'y1':np.array(y1)})

X = dane[["x1", "x2"]]
y2 = dane["y1"]
# model without intercept
reg = linear_model.LinearRegression(fit_intercept=False)
# weighted
reg.fit(X,y1,sample_weight=w)
#parameters
print(reg.coef_)
#array([ 2.32945022,  2.10992809])

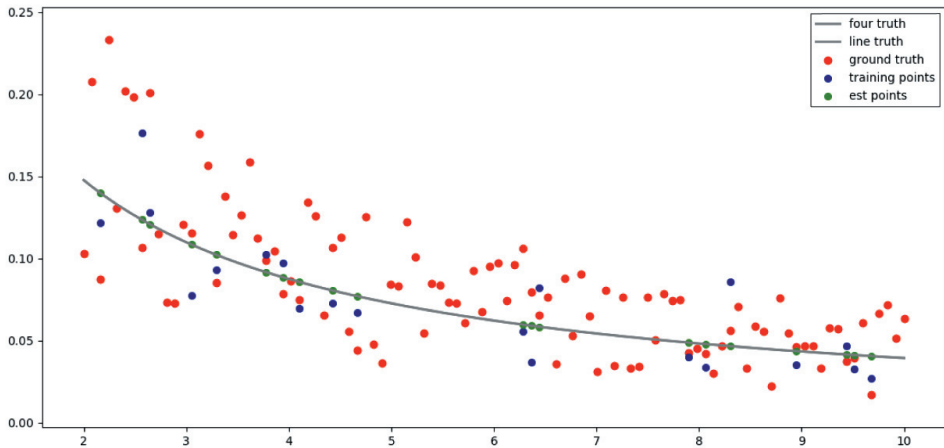
# not weighed
reg.fit(X,y1)
print(reg.coef_)
#array([ 1.72875482,  1.63686237])

```

```

plt.scatter(x_plot, h(x_plot), color='red',label="ground truth")
plt.scatter(x, y, color='navy', s=30, marker='o', label="training points")
y_est = 1/(2.3295*x+2.1099)
y_plot = 1/(2.3295*x_plot+2.1099)
plt.scatter(x, y_est, color='green', s=30, marker='o', label="est points")
plt.plot(x_plot, y_plot, color = 'grey', linewidth=2, label="line truth")
plt.legend(loc='best')
plt.show()

```



```

#####
# Example 1.3.6
# nonlinear regression ver 2
#####

```

```

import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn import linear_model

def g(x):
    return x/(np.random.uniform(1.5,2.5,size = len(x))*x + 2 ) - 0.005*np.random.normal(0, 1,
        size=len(x))

# generate data
x_plot = np.linspace(2, 10, 100)

# generate points and keep a subset of them
x = np.linspace(2, 10, 100)
rng = np.random.RandomState(0)
rng.shuffle(x)

```

```

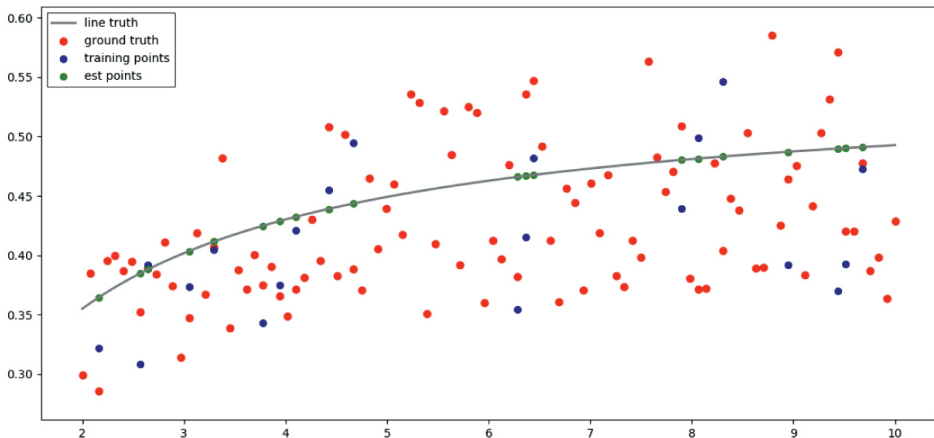
x = np.sort(x[:20])
y = g(x)

# the linear regression was used
x1 = y
x2 = x * y
y1 = x
w = y/x

dane = pd.DataFrame({
    'x2': np.array(x2),
    'x1': np.array(x1),
    'y1': np.array(y1)
})

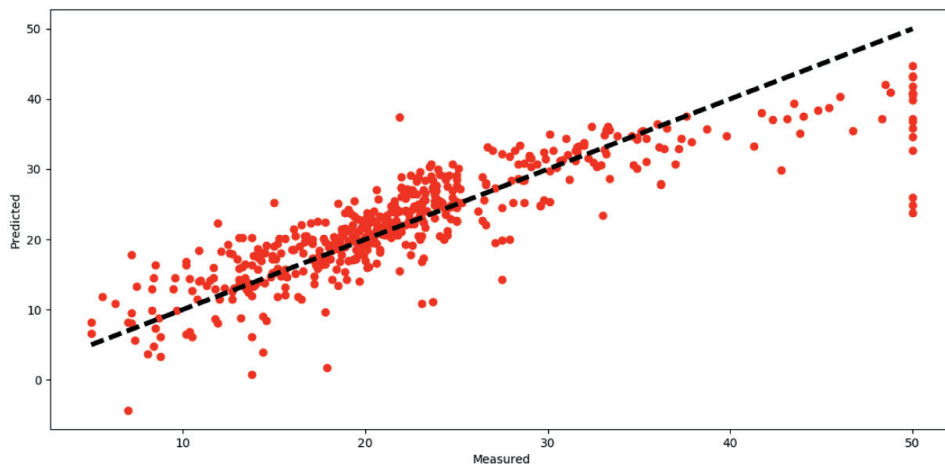
X = dane[["x1", "x2"]]
y2 = dane["y1"]
# model without intercept
reg = linear_model.LinearRegression(fit_intercept=False)
# weighted
reg.fit(X,y1,sample_weight=w)
#parameters
print(reg.coef_)
#array([ 1.83286441,  1.96808775])
plt.scatter(x_plot, g(x_plot), color='red',label="ground truth")
plt.scatter(x, y, color='navy', s=30, marker='o', label="training points")
y_est = x/(1.8329*x+1.968)
y_plot = x_plot/(1.8329*x_plot+1.968)
plt.scatter(x, y_est, color='green', s=30, marker='o', label="est points")
plt.plot(x_plot, y_plot, color = 'grey', linewidth=2, label="line truth")
plt.legend(loc='best')
plt.show()

```



```
#####  
# Example 1.3.7  
# linear regression for boston data  
#####
```

```
import matplotlib.pyplot as plt  
from sklearn import linear_model  
from sklearn import datasets  
  
reg = linear_model.LinearRegression()  
boston = datasets.load_boston()  
X = boston.data  
y = boston.target  
reg.fit(X,y)  
print(reg.coef_)  
print(reg.intercept_)  
plt.scatter(y, reg.predict(X), color='red',label="ground truth")  
plt.plot([y.min(), y.max()], [y.min(), y.max()], 'k--', lw=4)  
plt.xlabel('Measured')  
plt.ylabel('Predicted')  
plt.show()
```



## 2. Principle Component Analysis

### 2.1. The main idea of PCA

The previous section gave us a glimpse into the type of regression model which was required to define quantitative characteristics describing this model. And what if there is no information neither about qualitative, nor quantitative characteristics of regression? How should it look like in this case? The effective method of solving such tasks is the Principle Component Analysis – PCA.

PCA is one of the main way to reduce data dimension with minimum loss of information, developed by Karl Pearson in 1901. Basically, it is applied in many areas, such as: pattern recognition, computer sight, data compression, etc. PCA comes down to the calculation of eigenvectors and eigenvalues of the input data covariation matrix. Sometimes the PCA method is called the Hotelling transform (*the* Karhunen-Loeve Transform KLT) (see, for example, Bober et al. 2003). Enriching knowledge on PCA can be found for instance in Jolliffe, Li and Wang, Tanwar et. al., Tipping and Bishop, (see Hand 2011, Jolliffe 2002, Li, Wang 2005, Tanwar et al. 2018, Tipping, Bishop 1999).

Let's take the PCA method into account. At the beginning we will find the constant  $\mu$ , which in the best way describes input data

$$\varepsilon(\mu) = \sum_{i=1}^n (x_i - \mu)^2 \rightarrow \min_{\mu}.$$

In order to find the minimum we will equate the derivative to zero and find the value  $\mu$  delivering the minimum

$$\frac{d}{d\mu} \varepsilon(\mu) = -2 \sum_{i=1}^n (x_i - \mu) = 0 \Rightarrow \sum_{i=1}^n \mu = \sum_{i=1}^n x_i \Rightarrow \mu n = \sum_{i=1}^n x_i \Rightarrow \mu = \frac{1}{n} \sum_{i=1}^n x_i.$$

Furthermore, we will carry out mean-centering, that is, we will redefine original values as follows  $x_{\text{new}} = x_{\text{old}} - \mu$ , i.e. from each value  $x_i$  the mean  $\mu$  is subtracted. It is clear, that new data have the average equal to zero.

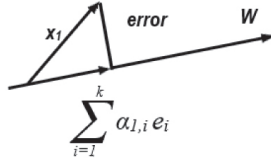
$$E(X - E(X)) = E(X) - E(X) = 0.$$

As a matter of fact, we made parallel translation in the existing coordinate system.

In the following section we assume that input data is centered. Not to mention that, we are going to find the most faithful representation of data  $D = \{x_1, \dots, x_n\}$  in some  $W$  subspace which has dimension  $k < n$ .

Let  $\{e_1, \dots, e_k\}$  be orthonormal basis of  $W$ . Any vector from  $W$  can be written in the form of linear combination of basis vectors, therefore for  $x_1$  it is possible to determine vector in the following form  $\sum_{i=1}^k \alpha_{1,i} e_i$ . The error between them (see Fig. 2.1) is calculated as follows

$$\varepsilon_1 = \left\| x_1 - \sum_{i=1}^k \alpha_{1,i} e_i \right\|_2^2 = \left\langle x_1 - \sum_{i=1}^k \alpha_{1,i} e_i, x_1 - \sum_{i=1}^k \alpha_{1,i} e_i \right\rangle.$$



**Fig. 2.1.** Vector error recovery illustration

To find the integral error, we need to sum the quantities errors over all  $x_j$ , so the total error is

$$\varepsilon(\underbrace{e_1, \dots, e_k, \alpha_{1,1}, \dots, \alpha_{n,k}}_{\text{unknowns}}) = \sum_{j=1}^n \varepsilon_j = \sum_{j=1}^n \left\| x_j - \sum_{i=1}^k \alpha_{j,i} e_i \right\|_2^2 \quad (2.1)$$

To minimize the error, it is necessary to calculate the partial derivative of an above mentioned function with respect to the coefficients and consider restrictions for orthogonality  $\{e_1, \dots, e_k\}$ . At the beginning, we will simplify the expression (2.1).



$$\begin{aligned}
\varepsilon(e_1, \dots, e_k, \alpha_{1,1}, \dots, \alpha_{n,k}) &= \sum_{j=1}^n \left\| x_j - \sum_{i=1}^k \alpha_{j,i} e_i \right\|_2^2 = \\
&= \sum_{j=1}^n \|x_j\|_2^2 - 2 \sum_{j=1}^n x_j^T \sum_{i=1}^k \alpha_{j,i} e_i + \sum_{j=1}^n \sum_{i=1}^k \alpha_{j,i}^2 = \\
&= \sum_{j=1}^n \|x_j\|_2^2 - 2 \sum_{j=1}^n \sum_{i=1}^k \alpha_{j,i} x_j^T e_i + \sum_{j=1}^n \sum_{i=1}^k \alpha_{j,i}^2.
\end{aligned}$$

Then

$$\frac{\partial}{\partial \alpha_{m,l}} \varepsilon(e_1, \dots, e_k, \alpha_{1,1}, \dots, \alpha_{n,k}) = -2x_m^T e_l + 2\alpha_{m,l}.$$

The necessary and sufficient condition for the extremum will take the following form

$$-2x_m^T e_l + 2\alpha_{m,l} = 0 \Rightarrow \alpha_{m,l} = x_m^T e_l.$$

Thus, the error (2.1) will be described in the form

$$\varepsilon(e_1, \dots, e_k) = \sum_{j=1}^n \|x_j\|_2^2 - 2 \sum_{j=1}^n \sum_{i=1}^k (x_j^T e_i) x_j^T e_i + \sum_{j=1}^n \sum_{i=1}^k (x_j^T e_i)^2.$$

After simplifying the equation (2.1), we receive

$$\varepsilon(e_1, \dots, e_k) = \sum_{j=1}^n \|x_j\|_2^2 - \sum_{j=1}^n \sum_{i=1}^k (x_j^T e_i)^2 \quad (2.2)$$

Taking into account that  $\langle a, b \rangle = a^T b$  and  $\langle b, a \rangle = \langle a, b \rangle$ , we obtain

$$(a^T b)^2 = (a^T b)(a^T b) = (b^T a)(a^T b) = b^T (aa^T) b,$$

therefore

$$\varepsilon(e_1, \dots, e_k) = \sum_{j=1}^n \|x_j\|_2^2 - \sum_{i=1}^k e_i^T \left( \sum_{j=1}^n (x_j x_j^T) \right) e_i = \sum_{j=1}^n \|x_j\|_2^2 - \sum_{i=1}^k e_i^T S e_i,$$

where  $S = \sum_{j=1}^n (x_j x_j^T)$  is the covariation matrix.

Next, the error  $\varepsilon(e_1, \dots, e_k) = \sum_{j=1}^n \|x_j\|_2^2 - \sum_{i=1}^k e_i^T S e_i$  will be minimized under the condition  $e_i^T e_i = 1$  for all  $i$ . Using the method of indefinite Lagrange multipliers, we enter multipliers  $\lambda_1, \dots, \lambda_k$  and, noticing that  $\sum_{j=1}^n \|x_j\|_2^2 \equiv \text{const}$ , we can describe the objective function

$$\lambda(e_1, \dots, e_k) = \sum_{i=1}^k e_i^T S e_i - \sum_{i=1}^k \lambda_i (e_i^T e_i - 1).$$

It is worth pointing out that  $(d/dX) \cdot (X^T X) = (d/dX) \cdot \langle X, X \rangle = 2X$  and if  $A$  is a symmetric matrix, then  $(d/dX) \cdot (X^T A X) = 2AX$ . As the result, we get

$$\frac{\partial}{\partial e_m} \lambda(e_1, \dots, e_k) = 2S e_m - 2\lambda_m e_m = 0,$$

that is,  $S e_m = \lambda_m e_m$ . Thus, it is necessary to find the solution of the equation  $(S - \lambda I) e = 0$  (here  $I$  is an identity matrix) that determine  $\lambda_m$  as eigenvalues and  $e_m$  as eigenvectors of the covariation matrix of  $S$ .

In this case, the error takes the following form

$$\varepsilon(e_1, \dots, e_k) = \sum_{j=1}^n \|x_j\|_2^2 - \sum_{i=1}^k \lambda_i \|e_i\|_2^2 = \sum_{j=1}^n \|x_j\|_2^2 - \sum_{i=1}^k \lambda_i \quad (2.3)$$

Minimization (2.3) consists in the selection  $k$  of the greatest eigenvalues and its corresponding eigenvectors of covariance matrix  $S$ . The bigger eigenvalue of the matrix  $S$  gives the bigger variation in the direction to the corresponding eigenvector. This result can be reformulated as follows – the projection  $X$  on the  $k$  – dimension subspace provides the greatest variation. Thus, the PCA can be treated as follows: we take orthogonal basis and rotate it on one of the directions as long as we do not receive the maximum variation. We fix this direction and we rotate the others, until we find the second direction and so on.

Let  $\{e_1, \dots, e_n\}$ , all eigenvectors of  $S$  matrix, be ordered in respect to the corresponding eigenvalue, then for any

$$x_i = \sum_{j=1}^n \alpha_{i,j} e_j = \underbrace{\alpha_{i,1}e_1 + \dots + \alpha_{i,k}e_k}_{\text{approximation}} + \overbrace{\alpha_{i,k+1}e_{k+1} + \dots + \alpha_{i,n}e_n}^{\text{error}}.$$

The coefficients  $\alpha_{m,l} = x_m^T e_l$  are the coordinates of the main components, for the greater value  $k$  gives the best approximation. At the same time, the main components are ordered according to the degree of the importance, that is, more important at the beginning and less important at the end.

Let's look at the algorithm of PCA.

Let start with input data  $D = \{x_1^0, \dots, x_n^0\}$ , where each of vectors  $x_i^0$  has dimension of  $N$

1. Let's find the average  $\mu = (1/n) \sum_{i=1}^n x_i^0$ .
2. Let's subtract the average from each vector  $x_i = x_i^0 - \mu$ .
3. Let's find the covariation matrix  $S = \sum_{j=1}^n x_j x_j^T$ .
4. Let's calculate eigenvectors  $\{e_1, \dots, e_k\}$ , corresponding to the  $k$  greatest eigenvalues of  $S$ .
5. Let  $\{e_1, \dots, e_k\}$  form the matrix  $E = [e_1 \dots e_k]$ .
6. Then the closest approximation of  $x$  is  $E^T x$ .

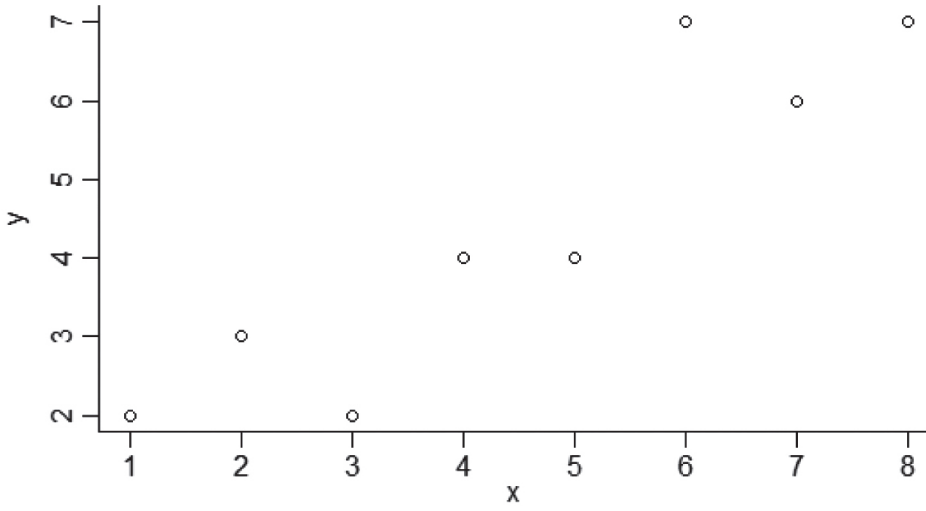
Let's review the example.

The set of data  $D^0 \left( (x_1^0, y_1^0), \dots, (x_8^0, y_8^0) \right)$  is determined by the Table 2.1. Its graphic illustration are points on the plane shown in Figure 2.2.

Let's find mean value  $\mu = (4.5, 4.375)$ , then after centering data  $D$  will take the form as mention in Table 2.2 and shown in Figure 2.3.

**Table 2.1**  
Input data

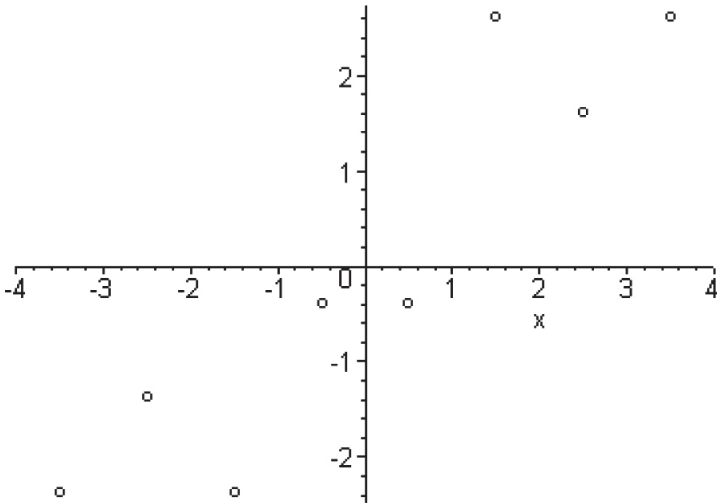
$x$	1	2	3	4	5	6	7	8
$y$	2	3	2	4	4	7	6	7



**Fig. 2.2.** The input data

**Table 2.2**  
The centered data

$x$	-3.5	-2.5	-1.5	-0.5	0.5	1.5	2.5	3.5
$y$	-2.375	-1.375	-2.375	-0.375	-0.375	2.625	1.625	2.625



**Fig. 2.3.** A parallel shift combining of origin data

Then:

$$s_{1,1} = \langle x, x \rangle = \sum_{i=1}^8 x_i x_i = 42,$$

$$s_{2,1} = s_{1,2} = \langle x, y \rangle = \sum_{i=1}^8 x_i y_i = 32.5,$$

$$s_{2,2} = \langle y, y \rangle = \sum_{i=1}^8 y_i y_i = 29.875,$$

and the covariation matrix can be written as follows

$$S = \begin{pmatrix} s_{1,1} & s_{1,2} \\ s_{2,1} & s_{2,2} \end{pmatrix} = \begin{pmatrix} 42 & 32.5 \\ 32.5 & 29.875 \end{pmatrix}.$$

Solving the equation

$$\begin{vmatrix} 42 - \lambda & 32.5 \\ 32.5 & 29.875 - \lambda \end{vmatrix} = 0 \Leftrightarrow (42 - \lambda)(29.875 - \lambda) - (32.5)^2 = 0,$$

we receive eigenvalues  $\lambda_1 = 68.99810959$ ,  $\lambda_2 = 2.876890413$ .

For determining the eigenvectors  $e_1 = (e_{1,1}, e_{1,2})^T$  and  $e_2 = (e_{2,1}, e_{2,2})^T$  let's find any uncommon solution of the following system:

$$\begin{cases} (s_{1,1} - \lambda_1)e_{1,1} + s_{1,2}e_{1,2} = 0, \\ s_{1,2}e_{1,1} + (s_{2,2} - \lambda_1)e_{1,2} = 0, \end{cases}$$

and, respectively, for second eigenvalues:

$$\begin{cases} (s_{1,1} - \lambda_2)e_{2,1} + s_{1,2}e_{2,2} = 0, \\ s_{1,2}e_{2,1} + (s_{2,2} - \lambda_2)e_{2,2} = 0. \end{cases}$$

Under the eigenvalue  $\lambda$  in the both system of equations the main determinant is equal zero. So in both cases the equations are linearly dependent. For finding the solutions it is necessary to take any nonzero values for first unknown and the second unknown determines from the one of equation respectively to each system of equations. Therefore for the example  $e_{1,1} = 1, e_{1,2} = 0.8307110643$  and  $e_{2,1} = 1, e_{2,2} = -1.203787987$ .

Thus, the vector  $e_1 = (1, 0.8307110643)^T$  corresponds to the eigenvalue  $\lambda_1 = 68.99810959$ , and the value  $\lambda_2 = 2.876890413$  corresponds the vector  $e_2 = (1, -1.203787987)^T$ . The bigger eigenvalue corresponds the greater principal direction. After normalizing eigenvectors to unit length, we get  $e_1 = (0.7692123649, 0.6389932223)^T$  and  $e_2 = (0.6389932223, -0.76922123648)^T$ .

It is necessary to determine the first main component  $z_1 = e_1^T D$  (see Tab. 2.3).

**Table 2.3**

The first main component

$z_1$	-4.20985	-2.80164	-2.67142	-0.62428	0.14498	2.83117	2.96139	4.3696
-------	----------	----------	----------	----------	---------	---------	---------	--------

Respectively, the second main component  $z_2 = e_2^T D$  will take the form as included within Table 2.4.

**Table 2.4**

The second main component

$z_2$	-0.4096	-0.53982	0.86839	-0.03104	0.60795	-1.0607	0.34751	0.21729
-------	---------	----------	---------	----------	---------	---------	---------	---------

Let's point out that for receiving a result for input data (uncentered) it is necessary to add the corresponding mean value.

The recovery of data referring to first main component (that is projections of input data to principal direction) will take the form  $x_i = e_{1,1}z_{1,i} + \mu_1, y_i = e_{1,2}z_{1,i} + \mu_2$  (see Tab. 2.5 and Fig. 2.4).

**Table 2.5**

The input data recovered on the first main the component

$0.769 \times z_1 + 4.5$	1.26174	2.3449	2.4451	4.0198	4.61158	6.6778	6.778	7.8611
$0.639 \times z_1 + 4.375$	1.6849	2.5848	2.668	3.9761	4.4676	6.1841	6.2673	7.1671

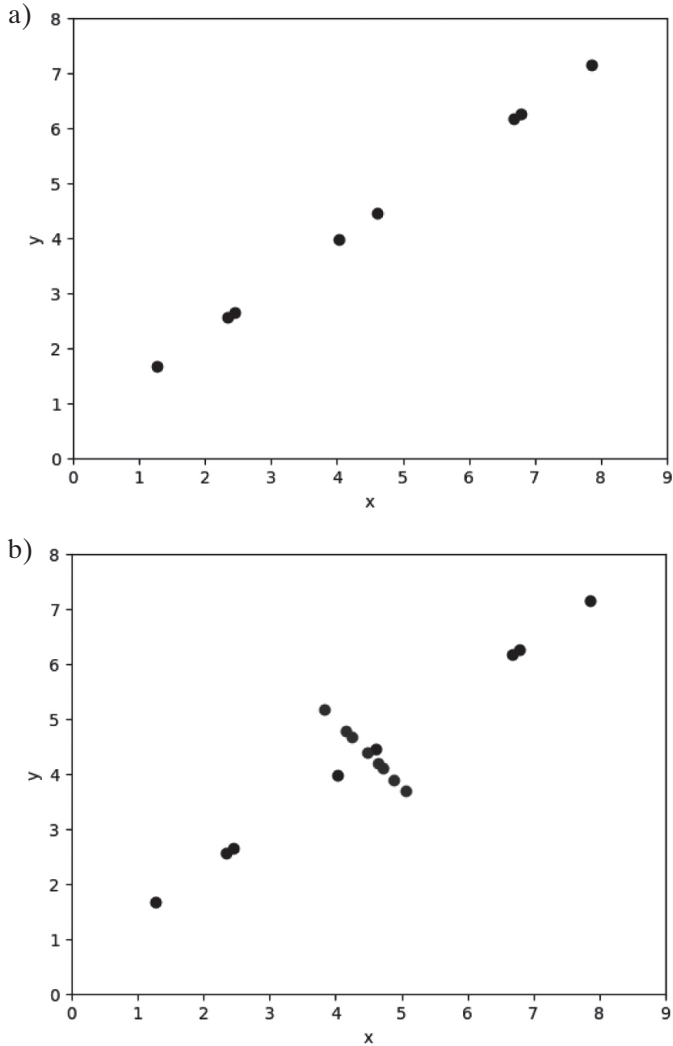


Fig. 2.4. The data presentation of the main components (in Python):  
a) the first component; b) the second component

## 2.2. An iteration scheme of PCA calculating

The described above method of determining the principal component is rather resource-intensive and unstable, especially if eigenvalues of the matrix are close to zero.

Basically, more effective is use of the iterative method of principal component. To achieve this aim, we can consider the task (2.1) from a different point of view.

For  $i=1$  case, the task (2.1) comes down to definition referring to the first component  $e_1$  which recovers all input data  $\{x_1, \dots, x_n\}$  the most efficiently.

$$\varepsilon(e_1, \alpha_{1,1}, \dots, \alpha_{n,1}) = \sum_{j=1}^n \|x_j - \alpha_{j,1}e_1\|_2^2 \rightarrow \min \quad (2.4)$$

on all  $e_1$  and  $\{\alpha_{i,1}\}_{i=1}^n$  under the condition  $\sum_{i=1}^n \alpha_{i,1}^2 = 1$ .

If  $\{\tilde{\alpha}_{i,1}\}_{i=1}^n$  and  $\tilde{e}_1$  is the solution of this task and  $\Delta x_j = x_j - \tilde{a}_{j,1}\tilde{e}_1$  - the error of data recovery based on the first main component, solving the following problem

$$\sum_{i=1}^n \|\Delta x_j - \alpha_{j,2}e_2\|_2^2 \rightarrow \min$$

on all  $e_2$  and  $\{\alpha_{i,2}\}_{i=1}^n$  under the condition  $\sum_{i=1}^n \alpha_{i,2}^2 = 1$ , we receive the second main component  $\tilde{e}_2$  and corresponding vector  $\{\tilde{\alpha}_{i,2}\}_{i=1}^n$  etc.

At fixed  $\{\alpha_{i,1}\}_{i=1}^n$  the problem (2.4) can be solved by the least-squares method. Knowing that the objective function represents the quadratic functional, the necessary and sufficient condition of the extremum are identical. Thus, the solution of the task comes down to solving a following equation

$$\frac{\partial}{\partial e_1} \varepsilon(e_1, \alpha_{1,1}, \dots, \alpha_{n,1}) = -2 \sum_{j=1}^n (x_j - \alpha_{j,1}e_1)\alpha_{j,1} = -2 \left( \sum_{j=1}^n x_j \alpha_{j,1} - \sum_{j=1}^n \alpha_{j,1}^2 e_1 \right).$$

From here we receive

$$e_1 = \frac{\sum_{j=1}^n x_j \alpha_{j,1}}{\sum_{j=1}^n \alpha_{j,1}^2}.$$

Considering the rationing condition unit, that is  $\sum_{i=1}^n \alpha_{i,1}^2 = 1$ , we get

$$e_1 = \sum_{j=1}^n x_j \alpha_{j,1}.$$



We can take the following step proceeding on the assumption that in the task (2.4) we know the component  $e_1$ . Also, it is required to find the extremum on  $\{\alpha_{i,1}\}_{i=1}^n$

$$\frac{\partial}{\partial \alpha_{v,1}} \varepsilon(e_1, \alpha_{1,1}, \dots, \alpha_{n,1}) = -2(x_v - \alpha_{v,1}e_1)e_1 = -2(\langle x_v, e_1 \rangle - \alpha_{v,1} \langle e_1, e_1 \rangle) = 0,$$

that is

$$\alpha_{v,1} = \frac{\langle x_v, e_1 \rangle}{\langle e_1, e_1 \rangle},$$

where, as usual  $\langle x, y \rangle$  is scalar product of vectors  $x$  and  $y$ .

Further, including known  $\{\alpha_{i,1}\}_{i=1}^n$ , we repeat all process. Moreover there will be no stabilization of the error yet. After getting  $e_1$  let's consider the first main component  $\tilde{e}_1$ . Then  $\Delta x_j = x_j - \tilde{\alpha}_{j,1}\tilde{e}_1$  - error recovery of data of the first main component.

Applying this algorithm to the error recovery  $\Delta x_j$ , we find the second main component  $e_2$  together with coefficients  $\alpha_{j,2}$ , etc.

Let's look at the algorithm of this method.

At the beginning we center data, subtracting the mean value from input data and further we assume that data are averagely equal to zero.

1. Let's put number of iteration  $v = 1$ .
2. We choose starting values  $\alpha_{j,2}$ , for example, let all of them be equal among themselves, that is  $\alpha_{i,1}^v = 1/\sqrt{n}$ ,  $i = 1, 2, \dots, n$ .
3. We calculate  $e_1^v = \sum_{j=1}^n x_j \alpha_{j,1}^v$ .
4. Further we find  $\beta_i = \langle x_i, e_1^v \rangle / \langle e_1^v, e_1^v \rangle$ , and, normalizing it to unit length, we receive

$$\alpha_{i,1}^{v+1} = \frac{\beta_i}{\sqrt{\sum_{j=1}^n \beta_j^2}}.$$

5. Next we take  $v = v + 1$ .
6. We perform the inspection of stop criterion i.e., as stabilization of coefficients  $\{\alpha_{i,1}^v\}_{i=1}^n$ , stabilization main components  $e_1^v$ , or in advance check the set fixed number of iterations. If the condition of the repetitive process is not satisfied, then we pass to point 3.

Let's illustrate the iteration scheme of looking for the principal components on the same example which is stated above.

For already centered data (see Table 2.2) we will give several iterations. So, let in the beginning  $v = 1$  and  $\alpha_{i,1}^1 = 1/\sqrt{2}$ ,  $i = 1, 2$ . Calculating  $e_{1,j}^1 = \alpha_{1,1}^1 x_j + \alpha_{2,1}^1 y_j$ , we receive the first approximation shown in Table 2.6.

**Table 2.6**  
First approximation the main component

$e_1^1$	-4.1542	-2.740	-2.740	-0.619	0.088	2.917	2.917	4.33
---------	---------	--------	--------	--------	-------	-------	-------	------

Further we will calculate  $\beta_i = \langle x_i, e_1^1 \rangle / \langle e_1^1, e_1^1 \rangle = (0.7697, 0.64447)$  and after the normalization we receive

$$\alpha_{i,1}^2 = \frac{\beta_i}{\sqrt{\beta_1^2 + \beta_2^2}} = (0.7667, 0.64343).$$

Thus, after the first iteration approximate values of input data will be equal  $\tilde{x}_i = \alpha_{1,1}^2 e_{1,i}^1 + \mu_1$ ,  $\tilde{y}_i = \alpha_{2,1}^2 e_{1,i}^1 + \mu_2$  (compare the result with Table 2.7).

**Table 2.7**  
The input data recovered on first approximation main components

$\tilde{x}$	1.3148	2.399	2.399	4.0256	4.5678	6.736	6.736	7.82
$\tilde{y}$	1.702	2.612	2.612	3.977	4.4319	6.2517	6.2517	7.172

After ten iterations, we receive  $\tilde{x}_i = \alpha_{1,1}^{11} e_{1,i}^{10} + \mu_1$ ,  $\tilde{y}_i = \alpha_{2,1}^{11} e_{1,i}^{10} + \mu_2$  (see Table 2.8 with Figure 2.5 and compare the result with Table 2.5 with Figure 2.4).

**Table 2.8**  
The input data recovered on the tenth iteration of approximating main components

$\tilde{x}$	1.2617	2.3449	2.445	4.02	4.6115	6.6778	6.7778	7.861
$\tilde{y}$	1.685	2.585	2.668	3.976	4.468	6.1841	6.2673	7.1671

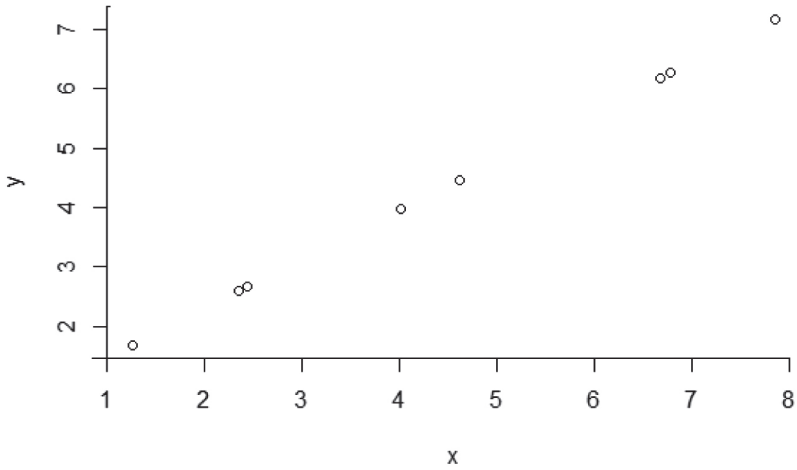


Fig. 2.5. Data presentation recovered on the tenth iteration of approximating main components

## 2.3. Examples in Python

### Example 1

```
#####
# Example 2.1.1
# PCA (example dataset)
# Data from table 2.1
#####
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# read data
x = [1,2,3,4,5,6,7,8]
y = [2,3,2,4,4,7,6,7]

plt.scatter(x,y,color='blue')
plt.xlabel('x')
plt.ylabel('y')

dane = pd.DataFrame({
    'x1': np.array(x),
    'y1': np.array(y),
})
```

```

X = dane[['x1','y1']]

cov_mat = (X-np.mean(X)).T.dot((X-np.mean(X)))
print('Covariance matrix \n%s' %cov_mat)

eig_vals, eig_vecs = np.linalg.eig(cov_mat)
print('Eigenvectors \n%s' %eig_vecs)
print('\nEigenvalues \n%s' %eig_vals)

components = eig_vecs.T.dot((X-np.mean(X)).T)
print('Main component \n%s' %components)

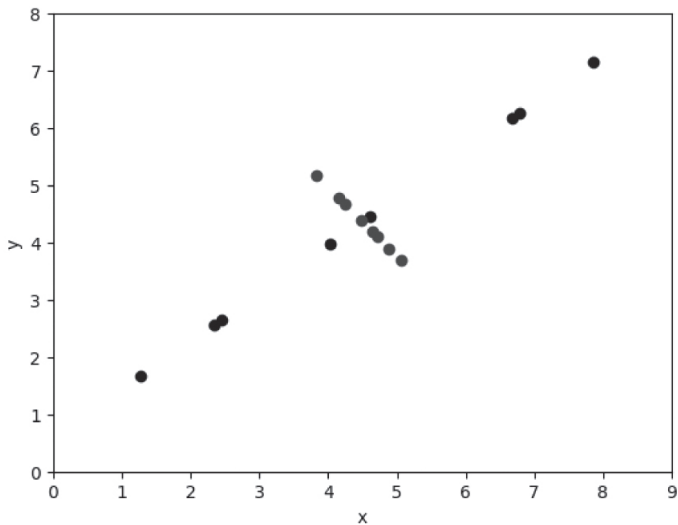
plt.close() # a new diagram
comp_11 = eig_vecs[0][0]*components[0] + np.repeat(np.mean(x),8)
comp_12 = eig_vecs[1][0]*components[0] + np.repeat(np.mean(y),8)

plt.scatter(comp_11,comp_12,color='blue')
plt.axis([0, 9, 0, 8])
plt.xlabel('x')
plt.ylabel('y')

comp_21 = eig_vecs[0][1]*components[1] + np.repeat(np.mean(x),8)
comp_22 = eig_vecs[1][1]*components[1] + np.repeat(np.mean(y),8)

plt.scatter(comp_21,comp_22,color='red')
plt.show()

```



## Example 2

```
#####  
# Example 2.1.2  
# PCA (example dataset)  
# Data from table 2.1  
#####  
import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt  
from sklearn import decomposition  
  
# read data  
x = [1,2,3,4,5,6,7,8]  
y = [2,3,2,4,4,7,6,7]  
dane = pd.DataFrame({  
    'x1': np.array(x),  
    'y1': np.array(y),  
})  
X = dane[['x1','y1']]  
  
pca = decomposition.PCA(n_components=2)  
pca.fit(X)  
  
Component = pca.fit_transform(X)  
print('Components: \n%s' %Component)
```

## Example 3

```
#####  
# Example 2.1.3  
# PCA (example dataset)  
# Dataset from table 2.2  
# Iterative method  
#####  
import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt  
  
# read data  
x = [1,2,3,4,5,6,7,8]  
y = [2,3,2,4,4,7,6,7]  
# data from table 2.2  
x1 = x - np.mean(x)  
y1 = y - np.mean(y)
```

```

dane = pd.DataFrame({
    'x1': np.array(x1),
    'y1': np.array(y1),
})
X = dane[['x1','y1']]

# declaration the empty variable
alfa = []
e = []
beta = []

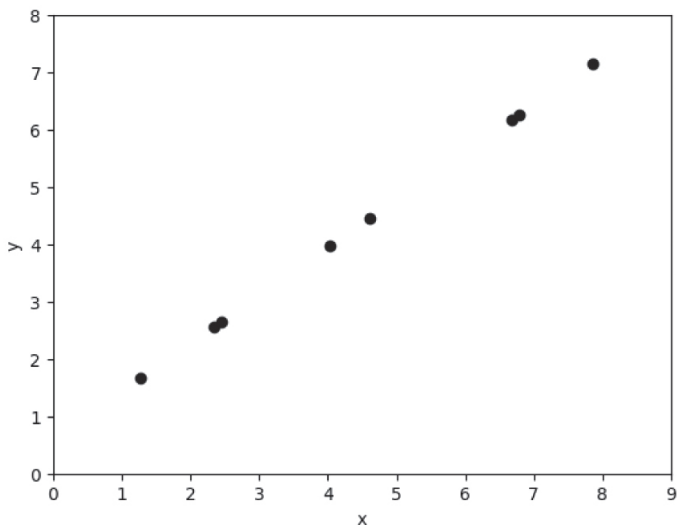
alfa.append( np.array([1/2**(0.5),1/2**(0.5)]))
e.append( alfa[0].dot(X.T) )

n = 10
for i in range(n):
    beta.append( np.array( [X['x1'].dot(e[i])/e[i].dot(e[i]),X['y1'].dot(e[i])/e[i].dot(e[i])] ) ) )
    alfa.append( beta[i]/(sum(beta[i]*beta[i])**0.5))
    e.append( alfa[i+1].dot(X.T) )

comp_1 = alfa[10][0]*e[10] + np.mean(x)
comp_2 = alfa[10][1]*e[10] + np.mean(y)

plt.scatter(comp_1,comp_2,color='blue')
plt.axis([0, 9, 0, 8])
plt.xlabel('x')
plt.ylabel('y')
plt.show()

```



## 2.4. Optimum transition from the RGB model to optimum three-component model

Let's have one example of use of the PCA in such area as computer graphics. Here outlining the transfer of the image from space of equal color characteristics into space of unequal ones is vital. All images are visualized with use of mixing equal color component – red, green and blue, which is called the RGB model. As unequal color component there are three components is used: value of illumination (the luminescent component), the characteristic of warm colors and the characteristic of cold shape. The avail unequal color component is used for compression of images and video flows and in each of the three components it uses the method of compression (see, for example, Archambea et al. 2008, Scholkopf et al. 1998, Tanwar et al. 2018, Idris 2014, Pedregosa et al. 2011).

It is possible to approach the problem of unequal color space creation from a different perspective, regarding the maximum informational content of everyone components. Let's apply the method of principal component to receive unequal three-component model of the root-mean-square error of recovery of the initial image, optimum from the point of view of minimization. That is, the first of the received color component will carry the most information about the image among all received color components, and the second will contain the most information among the remaining.

Thus, in our terminology, the task (2.1) will take the following form

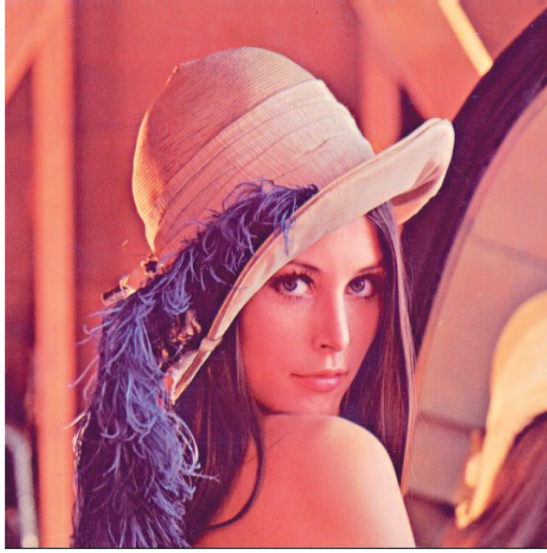
$$\left\| R - \sum_{i=1}^3 \alpha_{r,i} e_i \right\|_2^2 + \left\| G - \sum_{i=1}^3 \alpha_{g,i} e_i \right\|_2^2 + \left\| B - \sum_{i=1}^3 \alpha_{b,i} e_i \right\|_2^2 \rightarrow \min,$$

where the minimum undertakes on all  $\alpha_{r,i}$ ,  $\alpha_{g,i}$ ,  $\alpha_{b,i}$  and  $e_i$ ,  $i = 1, 2, 3$ .

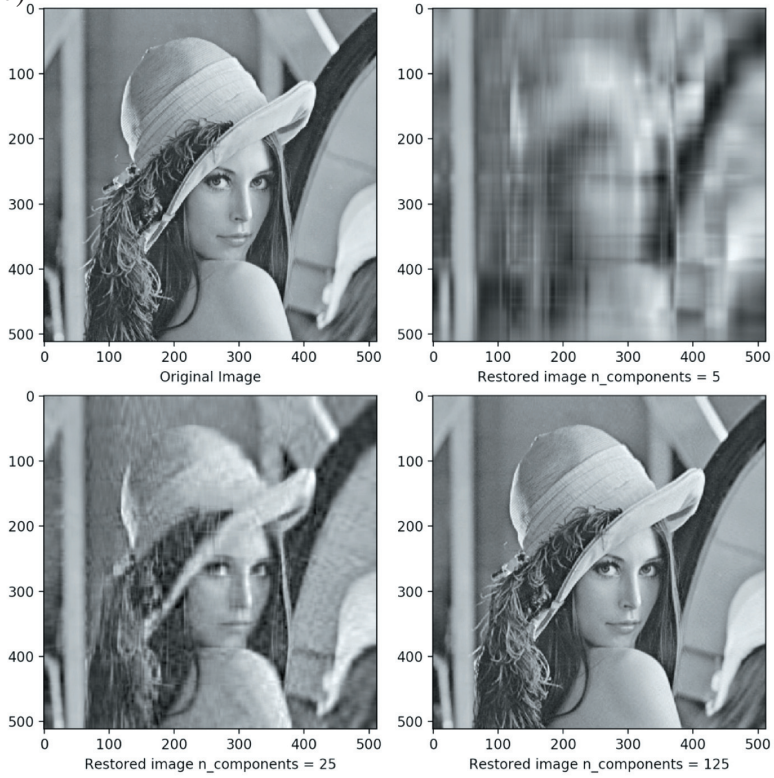
As data we can consider the test image Lena shown in Figure 2.6a. Applying the principal components method, we receive

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} 0.767785 & 0.45439 & 0.4517034 \\ -0.6164395 & 0.716085 & 0.3274513 \\ -0.6164395 & 0.716085 & 0.3274513 \end{pmatrix} \begin{pmatrix} e_1 \\ e_2 \\ e_3 \end{pmatrix}.$$

a)



b)



**Fig. 2.6.** Test image Lena (a); the image Lena in original state and restored in 5, 25, 125 components (b)



Here, in the first column of the matrix, there are coefficients  $\alpha_{r,1}$ ,  $\alpha_{r,2}$ ,  $\alpha_{r,3}$ , in the second  $\alpha_{g,1}$ ,  $\alpha_{g,2}$ ,  $\alpha_{g,3}$  and in the third  $\alpha_{b,1}$ ,  $\alpha_{b,2}$ ,  $\alpha_{b,3}$ . Then recovery of the test image after one component can be written down as follows:

$$R_{i,j} = 0.767785 \quad Y_{i,j},$$

$$G_{i,j} = 0.45439 \quad Y_{i,j},$$

$$B_{i,j} = 0.4517034 \quad Y_{i,j},$$

where  $Y_{i,j}$  – values of the first main components  $e_1$ , corresponding to pixel with coordinates  $(i, j)$ .

Also, we can compress an image using PCA without a significant loss of its variance. The earlier in this section we have demonstrated using PCA to compress high dimensional data to lower dimensional data. It is worth mentioning that PCA can also take the compressed representation of the data (lower dimensional data) back to an approximation of the original high dimensional data. If you are interested in the code that produces the image in Figure 2.6b, check out the example Python code given below.

```
#####
# Example 2.4.1
# Example for image reconstruction from compressed representation
# Principal Component Analysis in an image with Python, scikit-learn and scikit-image
#####

from sklearn.decomposition import PCA
from pylab import *
from skimage import data, io, color
import matplotlib.pyplot as plt
from matplotlib import gridspec

file = "Lenna.png"
lenna = io.imread(file, as_grey=True)
gs = gridspec.GridSpec(2, 2, width_ratios=[1, 1])

fig = plt.figure(figsize=(8, 8))
fig.subplots_adjust(hspace=0.4, wspace=0.4)

plt.subplot(gs[0])
io.imshow(lenna)
xlabel('Original Image')
```

```

for i in range(1, 4):
    n_comp = 5 ** i
    pca = PCA(n_components=n_comp)
    pca.fit(lenna)
    lenna_pca = pca.fit_transform(lenna)
    lenna_restored = pca.inverse_transform(lenna_pca)
    plt.subplot(gs[i])
    io.imshow(lenna_restored)
    xlabel('Restored image n_components = %s' % n_comp)
    print('Variance retained %s %%' % (
        (1 - sum(pca.explained_variance_ratio_) / size(pca.explained_variance_ratio_)) * 100))
    print('Compression Ratio %s %%' % (float(size(lenna_pca)) / size(lenna) * 100))

show()

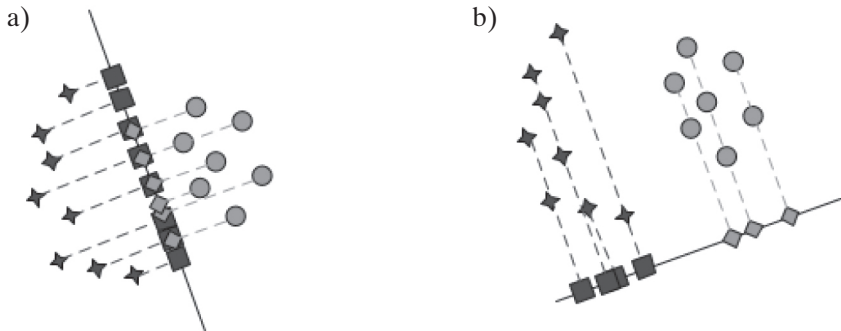
```

## 2.5. Fisher linear discriminant analysis

PCA is the most accurate representation of the data in the space of smaller dimension.

The Fisher linear discriminant analysis (FLDA) in its concept has a different nature. More information about FLDA the reader can find for example in the work Bober et al., Fisher, Iatan (see Bober et al. 2003, Fisher 1936, Iatan 2010, Pedregosa et al. 2011).

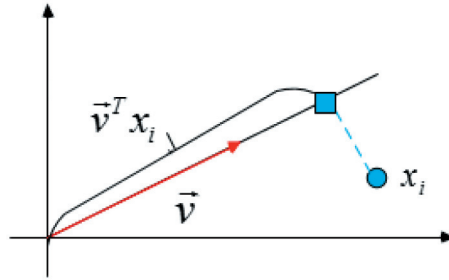
The main idea of FLDA is to find a hyperplane projection which allows to separate to separate classes of data the most accurately (see Fig. 2.7).



**Fig. 2.7.** Separation of data classes: a) a poor separation of point classes; b) a good separation of point classes

Suppose we have points  $x_1, x_2, \dots, x_n$  of the two classes in the measurements  $d$ , of which  $n_1$  points belong to the same class and  $n_2$  to the second one.

For the unit vector  $\vec{v}$  we can construct projections of these points on the direction of the vector (see Fig. 2.8).



**Fig. 2.8.** The projection of a point on the direction of a vector

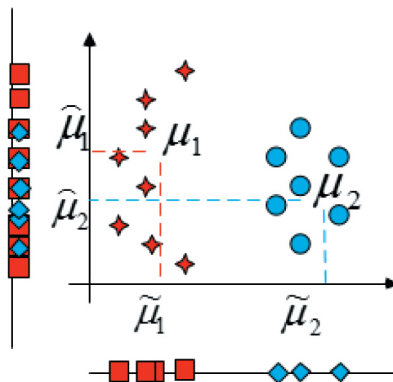
Then, the scalar product  $\vec{v}^T x_i$  coincides with the distance from the origin to the point of projection  $x_i$  in the direction of the vector  $\vec{v}$ , in other words,  $\vec{v}^T x_i$  is a projection  $x_i$  to the space of lower dimension.

We estimate the separation degree of the projections of different classes. Let  $\mu_1$  and  $\mu_2$  be the average values of the first and second classes, and  $\tilde{\mu}_1, \tilde{\mu}_2$  be the mean values of the projections of the first and second classes, then

$$\tilde{\mu}_1 = \frac{1}{n_1} \sum \{ \vec{v}^T x_i \mid x_i \in c_1 \} = \vec{v}^T \left( \frac{1}{n_1} \sum \{ x_i \mid x_i \in c_1 \} \right) = \vec{v}^T \mu_1$$

and correspondingly,  $\tilde{\mu}_2 = \vec{v}^T \mu_2$ .

A value  $|\tilde{\mu}_1 - \tilde{\mu}_2|$  can be a good measure to separate classes (see Fig. 2.9).



**Fig. 2.9.** Value  $|\tilde{\mu}_1 - \tilde{\mu}_2|$  as a measure of the separation of data classes

It's worth mentioning that, the horizontal axis for a given image separates the classes, more efficiently, because

$$|\tilde{\mu}_1 - \tilde{\mu}_2| > |\mu_1 - \mu_2|.$$

On the other hand, this characteristic is shared by only classes centers, that is not always the good separation of the classes themselves.

We need to normalize  $|\tilde{\mu}_1 - \tilde{\mu}_2|$  by a factor which is proportional to the scatter of the data class.

So, let the data be available  $z_1, z_2, \dots, z_n$ . Their average value can be calculated by the formula  $\mu_z = (1/n) \sum_{i=1}^n z_i$  and their variance can be determined by  $s = \sum_{i=1}^n (z_i - \mu_z)^2$ .

Thus, the spread is simply equal to the product of the variance by  $n$  (the number of samples), that is, the dispersion from scatter is different only in the scale. Fisher suggested the following decision i.e. normalization  $|\tilde{\mu}_1 - \tilde{\mu}_2|$  by data dispersion.

Let  $y_i = \vec{v}^T x_i$  be the projection of  $x_i$  to the direction of the vector  $\vec{v}$ .

Then scatter projections of the first class is  $\tilde{s}_1^2 = \sum \{(y_i - \tilde{\mu}_1)^2 \mid x_i \in c_1\}$  and second  $\tilde{s}_2^2 = \sum \{(y_i - \tilde{\mu}_2)^2 \mid x_i \in c_2\}$ .

The jointly normalization for both classes is carried out. Then Fisher's linear discriminant is a hyperplane whose direction  $\vec{v}$  maximizes the value

$$J(\vec{v}) = \frac{|\tilde{\mu}_1 - \tilde{\mu}_2|^2}{\tilde{s}_1^2 + \tilde{s}_2^2}.$$

All we need to do now is to express the  $J$  explicitly as a function of  $\vec{v}$  and maximize its value.

We define matrices for estimating the spread of the initial data (to the projection)

$$S_1 = \sum \{(x_i - \mu_1)(x_i - \mu_1)^T \mid x_i \in c_1\} \text{ and } S_2 = \sum \{(x_i - \mu_2)(x_i - \mu_2)^T \mid x_i \in c_2\}.$$

Now we can define the scattering matrix between classes  $S_w = S_1 + S_2$ .

Hence, noting that  $\tilde{s}_1^2 = \sum \left\{ (y_i - \tilde{\mu}_1)^2 \mid x_i \in c_1 \right\}$  and taking into account  $y_i = \tilde{v}^T x_i$  and  $\tilde{\mu}_1 = \tilde{v}^T \mu_1$  we get

$$\begin{aligned} \tilde{s}_1^2 &= \sum \left\{ \left( \tilde{v}^T x_i - \tilde{v}^T \mu_1 \right)^2 \mid x_i \in c_1 \right\} = \sum \left\{ \left( \tilde{v}^T (x_i - \mu_1) \right)^T \left( \tilde{v}^T (x_i - \mu_1) \right) \mid x_i \in c_1 \right\} = \\ &= \sum \left\{ \left( (x_i - \mu_1)^T \tilde{v} \right)^T \left( (x_i - \mu_1)^T \tilde{v} \right) \mid x_i \in c_1 \right\} = \\ &= \sum \left\{ \tilde{v}^T \left( (x_i - \mu_1) (x_i - \mu_1)^T \right) \tilde{v} \mid x_i \in c_1 \right\} = \tilde{v}^T S_1 \tilde{v}. \end{aligned}$$

Similarly,  $\tilde{s}_2^2 = \tilde{v}^T S_2 \tilde{v}$  and correspondingly,  $\tilde{s}_1^2 + \tilde{s}_2^2 = \tilde{v}^T S_1 \tilde{v} + \tilde{v}^T S_2 \tilde{v} = \tilde{v}^T S_w \tilde{v}$ .

We can define the scattering matrix between classes  $S_B = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T$ . Besides

$$|\tilde{\mu}_1 - \tilde{\mu}_2|^2 = \left( \tilde{v}^T \mu_1 - \tilde{v}^T \mu_2 \right)^2 = \tilde{v}^T (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T \tilde{v} = \tilde{v}^T S_B \tilde{v}.$$

With these constructions, the objective function can be written as follows

$$J(v) = \frac{|\tilde{\mu}_1 - \tilde{\mu}_2|^2}{\tilde{s}_1^2 + \tilde{s}_2^2} = \frac{\tilde{v}^T S_B \tilde{v}}{\tilde{v}^T S_w \tilde{v}}.$$

Then simply find the extremum of this function

$$\begin{aligned} \frac{d}{dv} J(v) &= \frac{\left( \frac{d}{dv} \tilde{v}^T S_B \tilde{v} \right) \tilde{v}^T S_w \tilde{v} - \left( \frac{d}{dv} \tilde{v}^T S_w \tilde{v} \right) \tilde{v}^T S_B \tilde{v}}{\left( \tilde{v}^T S_w \tilde{v} \right)^2} = \\ &= \frac{(2S_B \tilde{v}) \tilde{v}^T S_w \tilde{v} - (2S_w \tilde{v}) \tilde{v}^T S_B \tilde{v}}{\left( \tilde{v}^T S_w \tilde{v} \right)^2} = 0. \end{aligned}$$

Then equating the numerator of the fraction to zero we get

$$(S_B \bar{v}) \bar{v}^T S_w \bar{v} - (S_w \bar{v}) \bar{v}^T S_B \bar{v} = 0,$$

or what is the same

$$\frac{(S_B \bar{v}) \bar{v}^T S_w \bar{v} - (S_w \bar{v}) \bar{v}^T S_B \bar{v}}{\bar{v}^T S_w \bar{v}} = \frac{(S_B \bar{v}) \bar{v}^T S_w \bar{v}}{\bar{v}^T S_w \bar{v}} - \frac{(S_w \bar{v}) \bar{v}^T S_B \bar{v}}{\bar{v}^T S_w \bar{v}} = S_B \bar{v} - \lambda S_w \bar{v} = 0,$$

where  $\lambda = \frac{\bar{v}^T S_B \bar{v}}{\bar{v}^T S_w \bar{v}}$ .

Thus the discriminant search problem is reduced to the problem of finding the eigenvectors and eigenvalues, if  $S_w$  has full rank, i.e., there exist its inverse, then

$$S_B \bar{v} = \lambda S_w \bar{v} \Rightarrow S_w^{-1} S_B \bar{v} = \lambda \bar{v}.$$

For any vector  $x$ , consider  $S_B x$

$$S_B x = (\mu_1 - \mu_2) \left( (\mu_1 - \mu_2)^T x \right) = \alpha (\mu_1 - \mu_2) \text{ where } \alpha = (\mu_1 - \mu_2)^T x.$$

We obtain as a solution the sought eigenvalues and eigenvectors

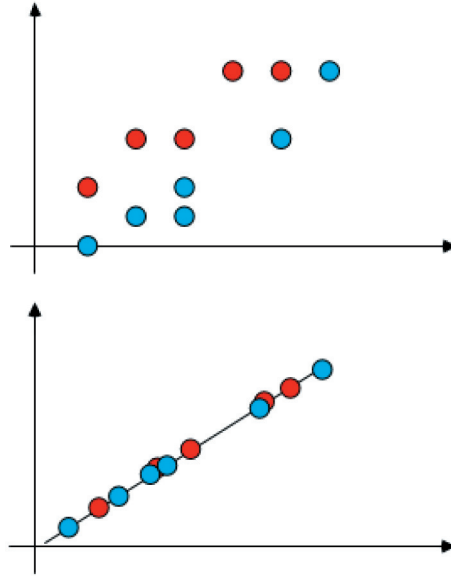
$$S_w^{-1} S_B \left( \underbrace{S_w^{-1} (\mu_1 - \mu_2)}_{\bar{v}} \right) = S_w^{-1} \left( \alpha (\mu_1 - \mu_2) \right) = \alpha \left( \underbrace{S_w^{-1} (\mu_1 - \mu_2)}_{\bar{v}} \right)$$

where  $\bar{v} = \left( S_w^{-1} (\mu_1 - \mu_2) \right)$ .

Let's consider an example.

Suppose there are two classes  $c_1 = [(1,2), (2,3), (3,3), (4,5), (5,5)]$  and  $c_2 = [(1,0), (2,1), (3,1), (3,2), (5,3), (6,5)]$  (see Fig. 2.10).

PCA allows you to get directions with the largest spread of data, which does not allow to split classes.



**Fig. 2.10.** The graphic interpretation of classes from example

At first, we find the average values  $\mu_1 = [3, 3.6]$  and  $\mu_2 = [3.3, 2]$ . We find the scatter matrix

$$S_1 = 4 \cdot \text{cov}(c_1) \begin{bmatrix} 10 & 8.0 \\ 8.0 & 7.2 \end{bmatrix} \quad S_2 = 4 \cdot \text{cov}(c_2) \begin{bmatrix} 17.3 & 16.0 \\ 16.0 & 16.0 \end{bmatrix}.$$

And for both classes

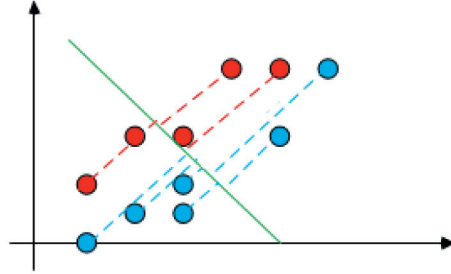
$$S_w = S_1 + S_2 = \begin{bmatrix} 27.3 & 24.0 \\ 24.0 & 23.2 \end{bmatrix}.$$

The matrix is invertible and

$$S_w^{-1} = \begin{bmatrix} 0.39 & -0.41 \\ -0.41 & 0.47 \end{bmatrix}.$$

The vector determining the optimal direction  $\bar{v}$  (see Fig. 2.11).

$$\bar{v} = S_w^{-1} (\mu_1 - \mu_2) = \begin{bmatrix} -0.79 \\ 0.89 \end{bmatrix}.$$



**Fig. 2.11.** The vector determining the optimal direction  $\vec{v}$

The final step is the calculation of one-dimensional vector  $y$ , and, respectively, the separation of classes:

$$Y_1 = \vec{v}^T c_1^T = [-0.65, 0.73] \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 3 & 5 & 5 \end{bmatrix} = [0.81 \dots 0.4],$$

$$Y_2 = \vec{v}^T c_2^T = [-0.65, 0.73] \begin{bmatrix} 1 & 2 & 3 & 3 & 5 & 6 \\ 0 & 1 & 1 & 2 & 3 & 5 \end{bmatrix} = [-0.65 \dots 0.25].$$

## 2.6. Multidimensional discriminant analysis (MDA)

MDA can be generalized for the case of several classes. In this case, the classes can be reduced to the dimension of 1, 2, 3, ...,  $s - 1$ . Let's consider the projection  $x_i$  on a linear subspace  $y_i = V^T x_i$ , where  $V$  is called the projection matrix.

Let  $n_i$  be the number of elements of the  $i$ -th class,  $\mu_i$  the average value of this class and  $\mu$  the average value of all elements. Then

$$\mu_i = \frac{1}{n_i} \sum \{x \mid x \in c_i\} \text{ and } \mu = \frac{1}{n} \sum_{i=1}^n x_i.$$

If we write the objective function

$$J(v) = \frac{\det(V^T S_B V)}{\det(V^T S_w V)},$$



then scatter matrix can look as follows

$$S_w = \sum_{i=1}^c S_i = \sum_{i=1}^c \sum \left\{ (x_k - \mu_i)(x_k - \mu_i)^T \mid x_k \in C_i \right\}$$

and the matrix of dispersion between classes can take the following form

$$S_B = \sum_{i=1}^c n_i (\mu_i - \mu)(\mu_i - \mu)^T.$$

Suppose that the matrix has the maximum rank =  $c - 1$ , then the problem is reduced to the solution of the eigenvalues equation

$$S_B V = \lambda S_w V.$$

Then the optimal projection would be on vector, which corresponds to the maximum eigenvalue.

### Parametric methods and discriminant functions

Using a parametric method, we assume availability of information on the form of the density of distribution of classes according to the known  $p_1(x | \theta_1)$ ,  $p_2(x | \theta_2)$ , ... where  $\theta_1, \theta_2, \dots$  are parameter estimation used in Bayesian classifier to separate classes. Using the linear discriminant functions  $\ell(\theta_1)$ ,  $\ell(\theta_2)$ , ... with parameters  $\theta_1, \theta_2, \dots$  also allows you to split data into classes.

Theoretically, the Bayesian classifier minimizes the risks, but in practice we do not have information about the density function. Fortunately, this is not so important for the separation of classes. In fact the exact definition of the density functions parameters is much more complicated than the estimation of exact discriminant functions. Therefore, the estimation of densities is often skipped. Naturally, the discriminant function may not be necessarily linear, but linear methods are quite popular and they should be considered, at least for the reason that the linear discriminant functions are optimal for Gaussian distributions with equal covariance.

Discriminant function is linear if it can be written as follows:  $g(x) = w^T x + w_0$ , where  $w$  is the weight vector and  $w_0$  is called an offset or threshold. The situation wherein the discriminant condition takes a form  $g(x) > 0 \Rightarrow x \in \text{Class [1]}$ ,  $g(x) < 0 \Rightarrow x \in \text{Class [2]}$ ,  $g(x) = 0 \Rightarrow x \in \text{bound}$  (see Fig. 2.12) is desirable.

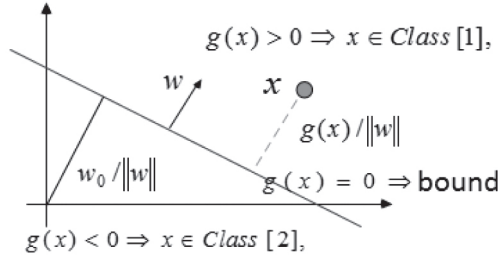


Fig. 2.12. The frontier class separation

Furthermore  $w$  determines the orientation of the separating hyperplane, and  $w_0$  places surface location solutions.

For the case of several classes the situation is similar. We can define  $m$  linear discriminant functions in the following form

$$g_i(x) = w_i^T x + w_{i0}, \quad i = 1, 2, \dots, m,$$

an element  $x$  belongs to the class  $c_i$ , if

$$g_i(x) \geq g_j(x), \quad \forall j \neq i.$$

This classifier is called a linear machine. A linear machine splits space into  $k$ -classes, and for elements of the  $i$ -th class the greatest value among all discriminant functions is named function  $g_i(x)$ . For two classes  $c_i, c_j$  the dividing line between these sets is hyperplane  $h_{ij}$  determined by the relation

$$g_i(x) = g_j(x) \Leftrightarrow w_i^T x + w_{i0} = w_j^T x + w_{j0} \Leftrightarrow (w_i - w_j)^T x + (w_{i0} - w_{j0}) = 0.$$

In this way, the vector  $w_i - w_j$  is normal to  $h_{ij}$  and the distance from  $x$  to  $h_{ij}$  is equal as follows

$$d(x, h_{ij}) = \frac{g_i(x) - g_j(x)}{\|w_i - w_j\|}.$$

In addition, the linear machine gives the decomposition into convex sets

$$x, y \in C_i \Rightarrow \alpha x + (1 - \alpha)y \in C_i.$$

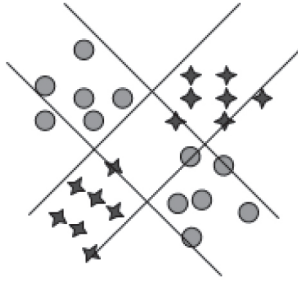
As the condition below is behaving

$$\forall j \neq i g_i(x) \geq g_j(x) \text{ and } g_i(y) \geq g_j(y) \Rightarrow$$

$$\forall j \neq i g_i(\alpha x + (1-\alpha)y) \geq g_j(\alpha x + (1-\alpha)y),$$

the result of the linear machines are simply connected domains.

Therefore, the applicability of the linear machine is primarily limited to conventional unimodal density  $p(x|\theta)$ . So for the sets shown on illustration below (see Fig. 2.13), the linear machine obviously cannot work well.



**Fig. 2.13.** The result of a linear machine for a simply connected domain

Let's rewrite a linear discriminant function  $g(x) = w^T x + w_0$  as

$$g(x) = \begin{bmatrix} w_0, w^T \end{bmatrix} \begin{bmatrix} 1 \\ x \end{bmatrix} = a^T y = g(y), \text{ where } a = \begin{bmatrix} w_0, w^T \end{bmatrix} \text{ and } y = \begin{bmatrix} 1 \\ x \end{bmatrix}.$$

Then if  $g(x) = w^T x + w_0$  for vector  $[x_1, \dots, x_d]^T$  the equivalent challenge can take the form  $g(y) = a^T y$  for vector  $[1, x_1, \dots, x_d]^T$  and the condition for the separation of classes can be written as follows

$$\begin{cases} a^T y_i > 0 & \forall y_i \in c_1, \\ a^T y_i < 0 & \forall y_i \in c_2, \end{cases} \text{ or what is the same } \begin{cases} a^T y_i > 0 & \forall y_i \in c_1, \\ a^T (-y_i) > 0 & \forall y_i \in c_2. \end{cases}$$

Thus, we need to find such a vector that for all elements  $y_i (i = 1, \dots, n)$  on one class it fulfills the condition

$$a^T y_i = \sum_{k=0}^d a_k y_i^{(k)} > 0.$$

There can be plenty of such vectors. Therefore we need to find a criterion for selecting the best one.

Anyway, whatever the criteria for selection are, it is necessary to find its extremum, say – a minimum. Below we give one fairly general algorithm for finding the extremum – the method of gradient descent.

Suppose you need to find the minimum of a function of several variables  $J(X) = J(x_1, x_2, \dots, x_d)$ . As you know, in order to find extremum, we need to calculate the derivative and equate it to zero. Here we determine the gradient and have the equation

$$\left[ \frac{\partial}{\partial x_1} J(X), \dots, \frac{\partial}{\partial x_d} J(X) \right]^T = \nabla J(X) = 0.$$

As it is known, the gradient is a vector which points the direction of greatest change of a function, respectively,  $-\nabla J(X)$  – is the direction of the greatest decrease.

Thus, in order to move from the point  $x^{(k)}$  to the point where the value of the objective function is lower, you need to shift to the gradient vector while the condition  $\eta^{(k)} \left| \nabla J(x^{(k)}) \right| > \varepsilon$  is satisfied, where  $\eta^{(k)}$  is parameter that regulates the speed and accuracy of the algorithm, that is

$$x^{(k+1)} = x^{(k)} - \eta^{(k)} \nabla J_p(a).$$

Unfortunately, the gradient descent can find a local extremum but not a global one.

### Perceptron principle

Let  $Y_M(a) = \{y_i : a^T y_i < 0\}$  be the set of incorrectly classified elements for the classifier with the vector  $a$ . Then the minimization of its quantity can be a criterion for the quality of the classifier.

Take the objective function  $J_p(a) = \sum \{(-a^T y) | y \in Y_M\}$ , then the condition is satisfied when misclassification  $a^T y < 0$ , i.e.  $J_p(a) \geq 0$ . There we will have  $\|a\|$  (just take the sum of the distances between misclassified elements and the border (it is shown in Fig. 2.14)).

$J_p(a)$  is a piecewise linear function, which is applicable to the method of gradient descent, then

$$\nabla J_p(a) = \sum \{(-y) | y \in Y_M\}.$$

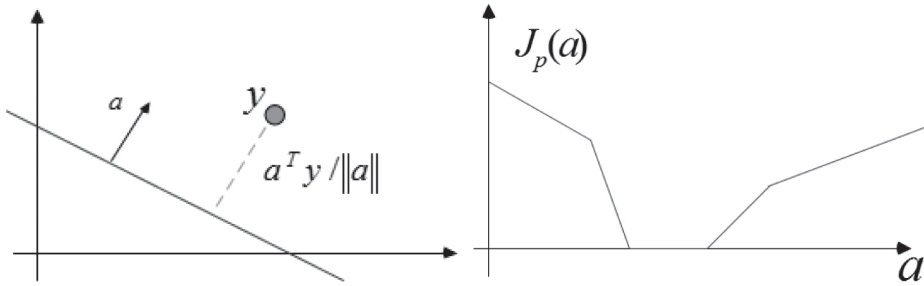


Fig. 2.14. Incorrect classification of elements

To make a step in the direction of the gradient

$$x^{(k+1)} = x^{(k)} - \eta^{(k)} \nabla J_p(a).$$

And calculate the new value of the vector at this point

$$a^{(k+1)} = a^{(k)} + \eta^{(k)} \sum \{y \mid y \in Y_M\}.$$

And for each incorrectly classified element we obtain

$$a^{(k+1)} = a^{(k)} + \eta^{(k)} y_M.$$

Geometric interpretation can be written as follows:  $(a^{(k)})^T y_M \leq 0$  indicating that the element  $y_M$  is not on the side of the separating hyperplane, and the addition of  $\eta^{(k)} y_M$  to a separating hyperplane moves in the right direction relative to the element  $y_M$ .

Let's consider the example. Table 2.9 contains the data for the example.

**Table 2.9**  
Incoming data

Name	Visit	Growth	In class sleeping	In the classroom chews the cud	Class
Peter	1 (true)	1 (true)	-1 (false)	-1 (false)	A
Lisa	1 (true)	1 (true)	1 (true)	1 (true)	F
John	-1 (false)	-1 (false)	-1 (false)	1 (true)	F
Diana	1 (true)	-1 (false)	-1 (false)	1 (true)	A

A linear discriminant function that separates the classes is determined.

At first we convert  $x_1, \dots, x_n$  to  $y_1, \dots, y_n$ . Next  $y_i \rightarrow -y_i \forall y_i \in c_2$  expanding the space on one dimension (see Tab. 2.10).

**Table 2.10**  
Determined extras

Name	Extras	Visit	Growth	In class sleeping	In the classroom chews the cud	Class
Peter	1	1 (true)	1 (true)	-1 (false)	-1 (false)	A
Lisa	-1	-1 (false)	-1 (false)	-1 (false)	-1 (false)	F
John	-1	1 (true)	1 (true)	1 (true)	-1 (false)	F
Diana	1	1 (true)	-1 (false)	-1 (false)	1 (true)	A

Assuming that  $\eta^{(k)} = 1$  we obtain  $a^{(k+1)} = a^{(k)} + y_M$ .

We choose  $a^{(1)} = [0.25, 0.25, 0.25, 0.25, 0.25]$  as starting vector and check the resulting classifier (see Tab. 2.11).

**Table 2.11**  
The result of the classifier

Name	$a^T y$	Classified incorrectly?
Peter	$0.25 \cdot 1 + 0.25 \cdot 1 + 0.25 \cdot 1 + 0.25 \cdot (-1) + 0.25 \cdot (-1) < 0$	false
Lisa	$0.25 \cdot (-1) + 0.25 \cdot (-1) + 0.25 \cdot (-1) + 0.25 \cdot (-1) + 0.25 \cdot (-1) < 0$	true
John	$0.25 \cdot (-1) + 0.25 \cdot 1 + 0.25 \cdot 1 + 0.25 \cdot 1 + 0.25 \cdot (-1) <? 0$	incorrect

Since we have misclassified item, we can modify the vector separating hyperplane  $a^{(k+1)} = a^{(k)} + y_M$

$$\begin{aligned}
 a^{(2)} &= a^{(1)} + y_M = [0.25, 0.25, 0.25, 0.25, 0.25] + [-1, -1, -1, -1, -1] = \\
 &= [-0.75, -0.75, -0.75, -0.75, -0.75]
 \end{aligned}$$

And we will continue checking (see Tab. 2.12).

**Table 2.12**  
The next result of classifier

Name	$a^T y$	Classified incorrectly?
John	$-0.75 \cdot (-1) - 0.75 \cdot 1 - 0.75 \cdot 1 - 0.75 \cdot 1 - 0.75 \cdot (-1) < 0$	true
Diana	$-0.75 \cdot 1 - 0.75 \cdot 1 - 0.75 \cdot (-1) - 0.75 \cdot (-1) - 0.75 \cdot 1 <? 0$	incorrect

Furthermore we modify

$$a^{(3)} = a^{(2)} + y_M = [-0.75, -0.75, -0.75, -0.75, -0.75] + [-1, 1, 1, 1, -1] = [-1.75, 0.25, 0.25, 0.25, -1.75]$$

and we can verify the last element in Table 2.13.

**Table 2.13**  
The last result of classifier

Name	$a^T y$	Classified incorrectly?
Diana	$-1.75 \cdot 1 + 0.25 \cdot 1 + 0.25 \cdot (-1) + 0.25 \cdot (-1) - 1.75 \cdot 1 <? 0$	incorrect

$$a^{(4)} = a^{(3)} + y_M = [-1.75, 0.25, 0.25, 0.25, -1.75] + [1, 1, -1, -1, 1] = [-0.75, 1.25, -0.75, -0.75, -0.75].$$

Finally we verify the quality of the resulting classifier in Table 2.14.

**Table 2.14**  
The quality of the resulting classifier

Name	$a^T y$	Classified incorrectly?
Peter	$-0.75 \cdot 1 + 1.25 \cdot 1 - 0.75 \cdot 1 - 0.75 \cdot (-1) - 0.75 \cdot (-1) < 0$	false
Lisa	$-0.75 \cdot (-1) + 1.25 \cdot (-1) - 0.75 \cdot (-1) - 0.75 \cdot (-1) - 0.75 \cdot (-1) < 0$	false
John	$-0.75 \cdot (-1) + 1.25 \cdot 1 - 0.75 \cdot 1 - 0.75 \cdot 1 - 0.75 \cdot (-1) < 0$	false
Diana	$-0.75 \cdot 1 + 1.25 \cdot 1 - 0.75 \cdot (-1) - 0.75 \cdot (-1) - 0.75 \cdot 1 < 0$	false

Thus the discriminant function has the following form

$$g(y) = -0.75y^{(0)} + 1.25y^{(1)} - 0.75y^{(2)} - 0.75y^{(3)} - 0.75y^{(4)}.$$

And, going back to the original notation

$$g(x) = 1.25x^{(1)} - 0.75x^{(2)} - 0.75x^{(3)} - 0.75x^{(4)} - 0.75.$$

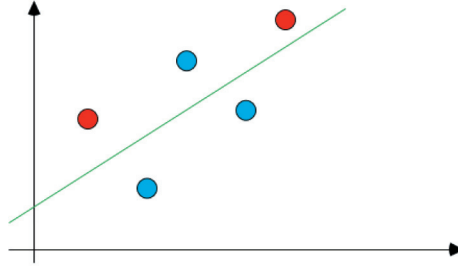
Element belongs to the class A, if

$$1.25x^{(1)} - 0.75x^{(2)} - 0.75x^{(3)} - 0.75x^{(4)} > 0.75$$

and to the class F if

$$1.25x^{(1)} - 0.75x^{(2)} - 0.75x^{(3)} - 0.75x^{(4)} < 0.75.$$

Unfortunately, the condition of linear separation of sets is far from always satisfied. Consider an example shown in Figure 2.15.



**Fig. 2.15.** The separation of sets on two data classes

There are two classes Class [1]: [2, 1], [4, 3], [3, 5] and Class [2]: [1, 3], [5, 6].

By analogy with the previous example, let's move to the new variables with the expansion of the space dimension

$$y_1 = [1, 2, 1]^T, y_2 = [1, 4, 3]^T, y_3 = [1, 3, 5]^T, y_4 = [-1, -1, -3]^T, y_5 = [-1, -5, -6]^T.$$

Following the algorithm Perceptron, choose initialize vector  $a^{(1)} = [1, 1, 1]$  and thus separating hyperplane  $x^{(1)} + x^{(2)} + 1 = 0$ .

We fix step  $\eta = 1$  and obtain the iterative solution  $a^{(k+1)} = a^{(k)} + y_M$ . We check the condition of separability performance classes:

$$y_1^T a^{(1)} = [1, 1, 1] \cdot [1, 2, 1]^T > 0 - \text{true},$$

$$y_2^T a^{(1)} = [1, 1, 1] \cdot [1, 4, 3]^T > 0 - \text{true},$$

$$y_3^T a^{(1)} = [1, 1, 1] \cdot [1, 3, 5]^T > 0 - \text{true},$$

$$y_4^T a^{(1)} = [1, 1, 1] \cdot [-1, -1, -3]^T = -5 < 0 - \text{false}.$$



We recalculate the normal vector separating hyperplane

$$a^{(2)} = a^{(1)} + y_M = [1, 1, 1] + [-1, -1, -3] = [0, 0, -2].$$

Then

$$y_5^T a^{(2)} = [0, 0, -2] \cdot [-1, -5, -5]^T = 12 > 0 - \text{true.}$$

Now, all over again

$$y_1^T a^{(2)} = [0, 0, -2] \cdot [1, 2, 1]^T < 0 - \text{false.}$$

We recalculate the vector again

$$a^{(3)} = a^{(2)} + y_M = [0, 0, -2] + [1, 2, 1] = [1, 2, -1].$$

And so on around the circle. It is clear that the algorithm converges.

You can change the perceptron learning step  $\eta^{(k)} = \eta^{(1)}/k$  to find a more accurate solution, if it exists. But it may not exist.

### Using the method of least squares

One of the most popular approaches is using the least squares method, the basic idea of which is to replace a system of inequalities by the system of equations.

If  $\|a\| = 1$  and  $a^T y_i = b_i$  is the distance from the point  $y_i$  on the hyperplane with normal vector  $a$ , then all points of our classes satisfy the following system of equations

$$\begin{cases} a^T y_1 = b_1 \\ \vdots \\ a^T y_n = b_n \end{cases} \Rightarrow \begin{bmatrix} y_1^{(0)} & y_1^{(1)} & \cdots & y_1^{(d)} \\ y_2^{(0)} & y_2^{(1)} & \cdots & y_2^{(d)} \\ \vdots & \vdots & \ddots & \vdots \\ y_n^{(0)} & y_n^{(1)} & \cdots & y_n^{(d)} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_d \end{bmatrix} = \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_d \end{bmatrix} \Leftrightarrow Ya = b.$$

If matrix  $Y$  is not degenerate, then  $a = Y^{-1}b$ . And if not, what then? Well, at least we need to find some approximation.

Let's  $\varepsilon = Ya - b$ , then we write the objective function as follows

$$J(a) = \|Ya - b\|^2 = \sum_{i=1}^n (a^T y_i - b_i)^2.$$

Hence

$$\begin{aligned}\nabla J(a) &= \left[ \frac{\partial}{\partial a_1} J(a), \dots, \frac{\partial}{\partial a_d} J(a) \right]^T = \frac{dJ(a)}{da} = \sum_{i=1}^n \frac{d}{da} (a^T y_i - b_i)^2 = \\ &= \sum_{i=1}^n 2(a^T y_i - b_i) \frac{d}{da} (a^T y_i - b_i) = \\ &= \sum_{i=1}^n 2(a^T y_i - b_i) y_i = 2Y^T (Ya - b) = 0 \Rightarrow Y^T Ya = Y^T b.\end{aligned}$$

The matrix  $Y^T Y$  is not degenerate so we get  $a = (Y^T Y)^{-1} Y^T b$ .

The LSM procedure equivalent let us find the hyperplane that corresponds to the available sample (usually called the learning sample) in the best way. But it can guarantee to find a solution only if  $Ya > 0$  that is, all the components of the vector  $Ya = [a^T y_1, \dots, a^T y_n]$  are positive. In fact  $Ya \approx Yb$  and  $Ya = [b_1 + \varepsilon_1, \dots, b_n + \varepsilon_n]^T$  and it is not necessary that all values  $\varepsilon_i$  are positive. Subsequently if  $b_i + \varepsilon_i > 0$  then we obtain the separation of the sets, and if not – (which is possible for elements which are located too close to the separating hyperplane), the separation of the sets is impracticable.

Thus, in the case of a linearly inseparable classes, the LSM does not give a separating hyperplane. There may be a temptation to draw a hyperplane so that the distance from the element to the separating hyperplane satisfies the condition, but this still does not lead to the desired result, Suppose that we use  $\beta b$  instead of  $b$ . Hence if  $a^*$  is the LSM solution of  $Ya = b$ , then  $Ya = \beta b$  by the LSM can be

$$\arg \min_a \|Ya - \beta b\|^2 = \arg \min_a \beta^2 Y \left( \frac{a}{\beta} \right) - b^2 \Rightarrow \arg \min_a Y \left( \frac{a}{\beta} \right) - b^2 = \beta a^*.$$

So for every element  $Ya$  which is less than zero,  $y_i^T a < 0$ , we get  $y_i^T (\beta a) < 0$ .

Under assumption  $b_1 = b_2 = \dots = b_n = 1$ , the LSM leads to the Fisher's linear discriminant. Moreover when the sample tends to infinity, we obtain Bayes method with distribution in the following form

$$g_B(x) = P(c_1 | x) - P(c_2 | x).$$

In order to evaluate the pros and cons of the LSM method, it is necessary to look at a few examples.

### Example 1

Class [1]: [6, 9], [5, 7]; Class [2]: [5, 9], [0, 4] (see Fig. 2.16).

Draw “normalization” by adding one dimension

$$y_1 = [1, 6, 9]^T, y_2 = [1, 5, 7]^T, y_3 = [-1, -5, -9]^T, y_4 = [-1, 0, -4]^T.$$

And, accordingly, we obtain the matrix

$$Y = \begin{bmatrix} 1 & 6 & 9 \\ 1 & 5 & 7 \\ -1 & -5 & -9 \\ -1 & 0 & -4 \end{bmatrix}, \quad \text{choose } b = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \quad \text{and applying LSM, we obtain}$$

$$a = [2.7, 1.0, -0.9]^T.$$

Furthermore we receive  $Ya = [0.4, 1.3, 0.6, 1.1]^T \neq [1, 1, 1, 1]^T$  and the condition  $Ya > 0$  gives separating hyperplane.

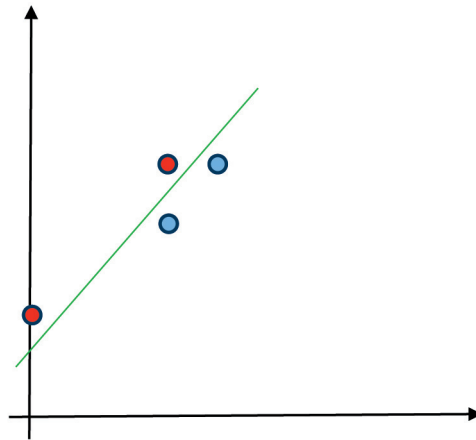


Fig. 2.16. The first example gives a separating hyperplane

### Example 2

Class [1]: [6, 9], [5, 7]; Class [2]: [5, 9], [0, 10] (see Fig. 2.17).

Draw normalization by adding one dimension

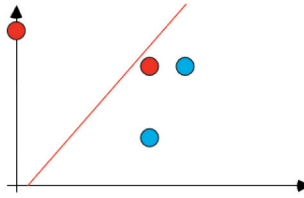
$$y_1 = [1, 6, 9]^T, y_2 = [1, 5, 7]^T, y_3 = [-1, -5, -9]^T, y_4 = [-1, 0, -10]^T.$$

And, accordingly, we obtain the matrix

$$Y = \begin{bmatrix} 1 & 6 & 9 \\ 1 & 5 & 7 \\ -1 & -5 & -9 \\ -1 & 0 & -10 \end{bmatrix}, \text{ choose } b = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \text{ and applying the LSM method, we obtain}$$

$$a = [3.2, 0.2, -0.4]^T.$$

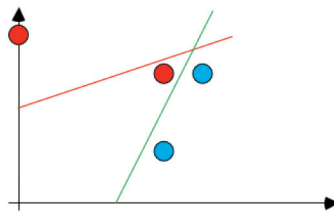
Moreover from this we get  $Ya = [0.2, 0.9, -0.04, 1.16]^T \neq [1, 1, 1, 1]^T$ , but  $a^T y_3 < 0$  and that does not allow receive the separating hyperplane.



**Fig. 2.17.** The second example does not give a separating hyperplane

Why do we get such a paradoxical result? Because the LSM is applied to all the values of the sample, while minimizing the total distance, it makes the method susceptible to “isolating points” and “noise”. But, using weights, i.e., giving the different elements different “importance” of their classification, the result can be corrected by getting an acceptable solution. Hence giving elements located far from separating hyperplane higher weights, we get the expected result.

Let’s choose  $b = [1, 1, 1, 10]^T$ . Applying LSM, we obtain  $a = [-1.1, 1.7, -0.9]^T$  and next we get  $Ya = [0.9, 1.0, 0.8, 10.0]^T \neq [1, 1, 1, 10]^T$  with result  $Ya > 0$  (see Fig. 2.18).



**Fig. 2.18.** The second example with weighting factors

Note that, for large-dimensional system of equations, the finding a solution can be difficult, especially under conditions where the value of the discriminant matrix is close to zero. In this case it is sufficient to start a search for an approximate solution, using the gradient descent method. So, there is the objective function

$J(a) = \|Ya - b\|^2 = \sum_{i=1}^n (a^T y_i - b_i)^2$ . Its gradient is  $\nabla J(a) = 2Y^T (Ya - b)$  and, following the slope of the gradient, we get each successive step of determining the weight vector  $a^{(k+1)} = a^{(k)} - \eta^{(k)} \nabla J(a) = a^{(k)} - \eta^{(k)} Y^T (Ya - b)$ .

If  $\eta^{(k)} = \eta^{(1)}/k$ , then to obtain the exact solution of the least squares method it is necessary to achieve the fulfillment of the condition  $Y^T (Ya - b) = 0$ . Descent algorithm always yields a solution, regardless of whether or not the matrix is invertible.

### Widrow–Hoff procedure

Widrow–Hoff procedure (see Widrow, Hoff 1988) is developed in relation to the “black box”, in which there are only linear relationships between inputs and outputs. The learning procedure is based on minimizing errors in the process of supplying input network of input images by using a gradient descent on the adjustable parameters of the neural network.

In our case  $a^{(k+1)} = a^{(k)} - \eta^{(k)} y_i (y_i^T a^{(k)} - b_i)$  so, a rule Widrow–Hoff (delta rule) provides vector correction  $a$  in the case when  $y_i^T a^{(k)}$  is not equal  $b_i$ .

### Ho-Kashyap procedure

The considered methods allow you to find the weight vector for linearly separable classes, and if the tolerance vector  $b$  is selected randomly, the result of the least squares method will be the minimization of expression  $\|Ya - b\|^2$ . If the classes are separable, there exist  $a$  and  $b$ , so that  $Ya = b > 0$ . The problem is that  $b$  is not known in advance. Ho-Kashyap procedure (see more in Theodoridis, Koutroumbas 2006, Simon 1986) involves the simultaneous finding of both the separating vector  $a$  and the tolerance vector  $b$ . The idea is, that if the sample (classes) are separable, then the minimum value  $J(a) = \|Ya - b\|^2$  is zero and the vector  $a$ , at which this value is reached, will be a separating vector.

As the necessary conditions for an extremum function of two variables, we get

$$\begin{cases} \nabla_a J(a, b) = 2Y^T (Ya - b) = 0, \\ \nabla_b J(a, b) = -2(Ya - b) = 0. \end{cases}$$

Ho-Kashyap procedure suggests using two-step algorithm for solving this problem.

1. For any fixed  $b$  from the first equation we find  $a$ .
2. For this fixed  $a$  from the second equation we get  $b$ .

The first step involves using a pseudo-inverse matrix, that is, from the following condition

$$2Y^T(Ya-b)=0 \text{ we can get } a=(Y^TY)^{-1}Y^Tb.$$

The second step involves receiving  $b$  for a fixed  $a$ , which seems to be the same as the  $b = Ya$ , but  $b$  must be positive, so to find  $b$  we apply the gradient descent method

$$b^{(k+1)} = b^{(k)} - \eta^{(k)} \nabla_b J(a^{(k)}, b^{(k)}).$$

For small values  $b$  and great gradient we still can receive a negative value  $b$ . To solve this problem, if the gradient has a large positive value, we assume that  $\eta$  is equal to 0.

$$b^{(k+1)} = b^{(k)} - \eta \frac{1}{2} \left( \nabla_b J(a^{(k)}, b^{(k)}) - \left| \nabla_b J(a^{(k)}, b^{(k)}) \right| \right).$$

Let  $\varepsilon^{(k)} = Ya^{(k)} - b^{(k)} = -\frac{1}{2} \nabla_b J(a^{(k)}, b^{(k)})$  be an error value, then

$$b^{(k+1)} = b^{(k)} - \eta \frac{1}{2} \left( -2\varepsilon^{(k)} - \left| 2\varepsilon^{(k)} \right| \right) = b^{(k)} + \eta \left( \varepsilon^{(k)} + \left| \varepsilon^{(k)} \right| \right).$$

Let's formalize Ho-Kashyap procedure.

Suppose, at first,  $k = 1$  and for any starting values  $a^{(1)}, b^{(1)}$  the procedure performs the following steps:

1.  $\varepsilon^{(k)} = Ya^{(k)} - b^{(k)}$ .
2.  $b^{(k+1)} = b^{(k)} - \eta \frac{1}{2} \left( -2\varepsilon^{(k)} - \left| 2\varepsilon^{(k)} \right| \right) = b^{(k)} + \eta \left( \varepsilon^{(k)} + \left| \varepsilon^{(k)} \right| \right)$ .
3.  $a^{(k+1)} = (Y^TY)^{-1} Y^T b^{(k+1)}$ .
4.  $k = k+1$  until the stop condition is fulfilled, which can be, for example, the limitation of the number of steps  $k$ , the stabilization of the vector  $b$  or the error value  $\varepsilon$ .

## 3. Application of fuzzy logic in Data Mining

### 3.1. What is fuzzy thinking?

AI problems (artificial intelligence) have been developed by mankind since time immemorial, from Galatea, when Pygmalion created to “Skynet”, was defeated by Schwarzenegger terminator. How can we distinguish the crafty AI from a normal person? Clearly, the answer to this question is included in Alan Turing article “Computing Machinery and Intelligence” from 1950. The proposed solution is known as the Turing test. But in June 6th, 2014 Virtual Boy known as “Eugene Gustman from Odessa” passed the Turing Test by convincing a group of people, via chat, that it was actually a 13-year-old-boy. The conversation was conducted in a way that machine’s answers seemed to belong to the man, not the machine. Why had this step taken so much time? And why did it happen?

Unlike a computer program, a person (expert), involved in solving the problem, usually relies on common sense, using vague and ambiguous terms. For example, an expert might say, “despite I’m a little tired, but I can still do the job for a while.” Other experts have no difficulty understanding and interpreting this statement. And what about IT-skilled computer providing the same level of understanding? Is it possible to develop such a program that the computer could use vague and ambiguous terms? Can we achieve it?

An instrument to achieve this aim is fuzzy logic. Specifically, the term relies on the theory of fuzzy sets, namely, sets that calibrate vagueness (ambiguity). Fuzzy logic is based on the idea that all concepts make gradation of possibilities. Boiling water is very hot, the tea is not enough strong, soup is insufficiently salted, weather conditions are not congenial. Such a gradient scale helps to distinguish the items belonging to certain class of components or these which do not.

Classical logic uses crisp differences. This is about drawing a line separating two sets. For example, according to the Beaufort scale, if the wind speed hits 10.7 m/s, then it gives fresh breeze, but if it hits approximately 10.8 m/s then it is already defined as a strong breeze.

Fuzzy logic reflects how people think, trying to simulate the feelings covered behind our words, our decisions and our common sense. As a result, it leads to the construction of words relativization more than in human intelligence systems if we may say so.

Fuzzy logic or many-valued logic was introduced in the 1930s, by Polish logician and philosopher Jan Lukasiewicz. Classical logic accepts only two values: 1 (true) and 0 (false), while Lukasiewicz logic expands the range of truth values to all real numbers between 0 and 1. This approach has led to inaccurate methods of reasoning, called “Theory opportunities”.

In 1965, Lotfi Zadeh, professor and head of the Department of Electrical Engineering, at University of California in Berkeley published his famous work “Fuzzy Sets” (Zadeh 1965). In fact, Zadeh renewed, collected and researched this phenomenon, translating the theory of opportunities in the formal system of mathematical logic and, more importantly, he introduced a new concept for the application of natural language. This new logic for representing and manipulating fuzzy terms was called fuzzy logic.

In contrast to the two-valued classical logic, fuzzy logic is ambiguous. It has something to do with the degree of compliance and degrees of truth. Fuzzy logic uses logical continuum of values between 0 (completely false) and 1 (fully right). So instead of just black and white (see Fig. 3.1a), it uses a range of colors (see Fig. 3.1b), assuming that the concept may be partly true and partly false simultaneously.

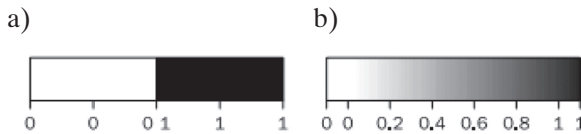


Fig. 3.1. The range of values for classical logic (a) and fuzzy logic (b)

### 3.2. Fuzzy sets

The concept of the set is fundamental for mathematics. Let  $X$  be a classic crisp set, and  $x$  an element. Then the element  $x$  belongs to  $X$  ( $x \in X$ ) or does not belong to  $X$  ( $x \notin X$ ), thus classical set theory imposes a crisp boundary and gives value 1 to each element of the set and value 0 to all elements that are not included within the prescribed set. Theory of distinct sets is determined by the logic that uses one of two values: true or false. The basic idea of fuzzy sets theory is that an element belongs to a fuzzy set with a certain degree of membership. Thus, the sentence is neither true nor false, but may be partly true (or partly false) to some extent. This degree is commonly accepted as a real number in the interval  $[0, 1]$ . More detailed explanation can be found



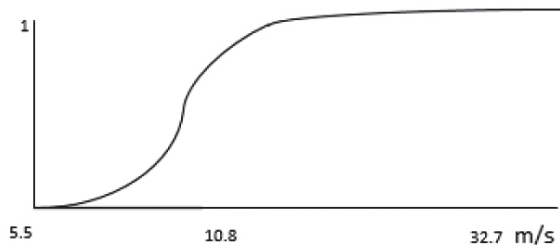
for instance in (Dubois and Prade 1988, Dubois 1993, Fedrizzi and Pasi 2008, Kaufmann 1975a, Kaufmann 1975b, Lee 1990, Pedrycz 1984, Skalna et al. 2015, Yager et al. 1987, Yager 1986, Yager and Zadeh 1992, Zadeh 1965, 1996).

So, a simple question: “How strong is the wind today?”, has a clear answer according to *Beaufort* force wind scale, the *Beaufort* strong wind begins with a speed of 10.8 m/s (see Fig. 3.2) so, if the wind speed is faster than 10.8 m/s, the wind is strong.



**Fig. 3.2.** A crisp set describing the strong wind

And going back to the question: “How strong is the wind today?”. In fuzzy terms the answer can be different. A fuzzy set is able to ensure a smooth transition through the boundary between the concepts of strong wind and light wind (see Fig. 3.3).



**Fig 3.3.** A fuzzy set describing the strong wind

It is possible to consider answers such as “not very strong wind”, “very light wind”, “calm wind” and so on.

A fuzzy set can be simply defined as a set with fuzzy boundaries.

Let  $X$  be the universe and its elements are designated as  $x$ . In classical set theory, a crisp subset  $A$  from  $X$  is defined by the function  $f_A(x)$  (the so-called characteristic function  $A$ )

$$f_A(x): X \rightarrow \{0,1\} \text{ where } f_A(x) = \begin{cases} 1, & \text{if } x \in A, \\ 0, & \text{if } x \notin A. \end{cases}$$

The result is a mapping of  $X$  into a set of two elements. For any element  $x$  of universe  $X$ , the characteristic feature  $f_A(x)$  is equal to 1 if  $x$  is an element of the set  $A$ , and equal to 0 if  $x$  is not an element  $A$ .

For the case of fuzzy set theory a membership function for set  $A$ , subset of universe  $X$ , is defined by function  $\mu_A(x)$  as follows:

$$\mu_A(x): X \rightarrow [0,1],$$

where:

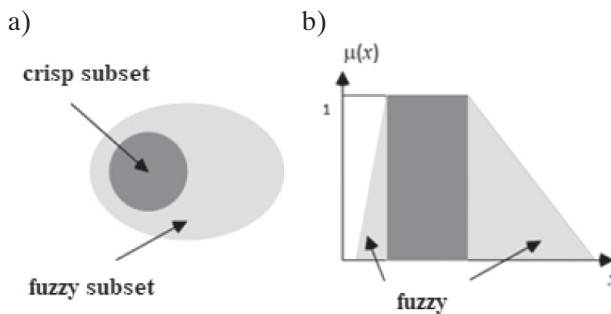
$$\mu_A(x) = 1 \text{ if } x \text{ is totally in } A,$$

$$\mu_A(x) = 0 \text{ if } x \text{ is not in } A,$$

$$0 < \mu_A(x) < 1 \text{ if } x \text{ lies partly in } A.$$

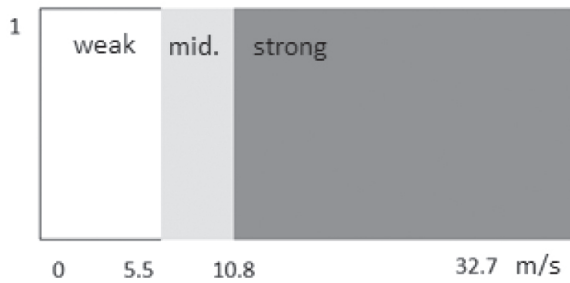
This set of axioms allows to consider a wide range of options. For any element  $x$  of the universe  $X$ , membership function  $\mu_A(x)$  is equal to the degree in which  $x$  is element of set  $A$ . This degree (a value of  $[0, 1]$ ) represents the degree of membership value which is also labeled as accessories of element  $x$  in the set  $A$ .

Following slightly more formal approach to the concept of fuzzy set (this is not possible without its implementation), we need to express it in a functional form, and then compare the elements of the set to their degree of affiliation (see Fig. 3.4). Typical forms are triangular and trapezoidal representations, as well as various straight and sigmoid ones. In practice, most applications use a linear function (of a triangular and trapezoidal shape).



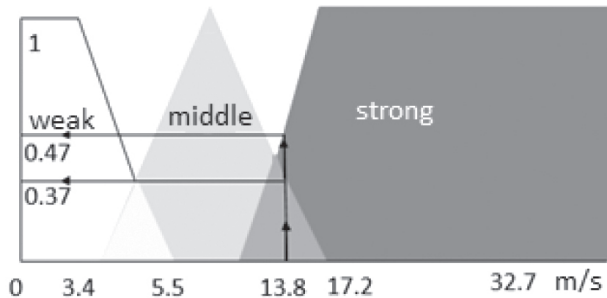
**Fig. 3.4.** An illustration of crisp set and fuzzy set (a); their membership functions (b)

According to the estimation of the wind force on the *Beaufort* scale, we can describe crisp sets representing the weak, medium and strong wind (see Fig. 3.5).



**Fig. 3.5.** A crisp set of the wind power classification

A similar fuzzy set can be represented as it is shown in Figure 3.6.



**Fig. 3.6.** A fuzzy set classification of the wind power

Note that in a fuzzy case the wind at a speed of 13.8 m/s belongs to “medium wind” with a level of 0.37, and also to “strong wind” with a degree of affiliation 0.47. Thus, the wind of 13.8 m/s has a partial membership in two or multiple sets.

Now, let  $X$  represents a universe of distinct set containing seven elements

$$X = \{x_1; x_2; x_3; x_4; x_5; x_6; x_7\}.$$

In addition, let  $A$  be a crisp subset  $X$  and consists of three elements:  $A = \{x_2; x_3; x_6\}$ . In this case, the subset  $A$  can be described as follows

$$A = \{(x_1, 0), (x_2, 1), (x_3, 1), (x_4, 0), (x_5, 0), (x_6, 1), (x_7, 0)\},$$

i.e., as a set of pairs  $\{(x_i, \mu_A(x_i))\}$  where  $\mu_A(x_i)$  is the degree of membership referring to element  $x_i$  belonging to a subset of  $A$ .

If  $X$  is the reference set and  $A$  is a subset of  $X$ , then  $A$  is called a fuzzy subset  $X$ , if and only if

$$A = \{(x, \mu_A(x))\}, x \in X, \mu_A(x) : X \rightarrow [0, 1].$$

In the particular case  $\mu_A(x) : X \rightarrow \{0, 1\}$ , the fuzzy subset  $A$  becomes crisp.

A fuzzy subset  $A$  is a finite support of set  $X$  represented as

$$A = \{(x_1, \mu_A(x_1)), (x_2, \mu_A(x_2)), \dots, (x_n, \mu_A(x_n))\}.$$

The subset  $A$  often takes the following form

$$A = \{\mu_A(x_1)/x_1; \mu_A(x_2)/x_2; \dots; \mu_A(x_n)/x_n\}.$$

Here, the delimiter “/” is used to associate membership values with its coordinate along the horizontal axis (abscissa).

As an example, we can write the following conformity vector:

Strong wind = (0/5.5; 0/10.8; 0.47/8.13; 1/17.2)

Average wind = (0/3.4; 1/8.0; 0.37/8.13; 0/17.2), etc.

### 3.3. Linguistic variables and linguistic gain

At the core of the fuzzy set theory is the idea of linguistic variables (see more in Mamdani, Assilian 1975, Mamdani 1977, Olaru, Wehenkel 2003, Skalna et al. 2015, Yager, Zadeh 1992, Zadeh 1975). Linguistic variable corresponds with fuzzy variable. For example, the statement “high rating” means that the linguistic variable “rating” has a high linguistic value. In fuzzy expert systems, linguistic variables are used to construct fuzzy rules. For example:

- IF the wind is strong THEN you should close window,
- IF it is getting dark THEN you should turn on the light.

The range of possible values of the linguistic variable can be quite varied, since the wind velocity may range from 0 to more than 32.7 m/s, which includes all subsets: calm, quiet wind, moderate wind, strong wind, storm, tornado, and others.

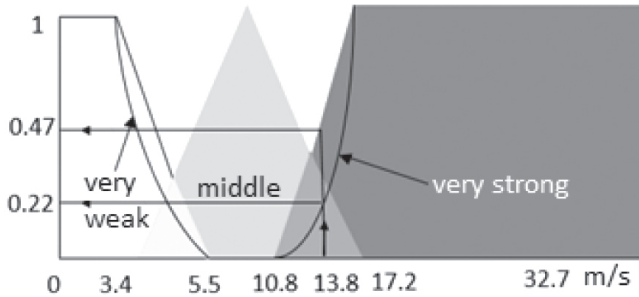
Added to this, there is a concept of linguistic amplification referring to terms that modify the shape of the fuzzy sets. They include adverbs such as: very, somewhat, rather, more or less, slightly, and so on.

Linguistic variables are used as:

- universal modifiers such as very, fairly, highly;
- true values such as true, completely, false, mainly;
- probability, for example, likely or not very likely;
- quantifiers, such as most, a few or a little;
- features such as almost impossible, or possible.

Linguistic amplification can effectively create new subsets. We can enlist a lot of meanings relating to “ripe tomato” which create a subset including, for instance “not very ripe tomatoes”, “not quite ripe tomatoes”, “very ripe tomatoes”, “more or less ripe tomatoes” and so on. At the same time the set of “more or less ripe tomatoes” is wider than just a set of “ripe tomatoes”. Thereby, using linguistic amplification can either extend or narrow the initial set.

Thus, linguistic amplification allows applications on the set of “strong wind” to allocate a subset of “very strong winds”. In this case, the wind at a speed of 13.8 m/s with a degree of membership 0.47 is a “strong wind” and with a degree of membership 0.22 is “very strong wind”, which is quite reasonable as it is shown in Figure 3.7.



**Fig. 3.7.** The example of linguistic hedges

For that matter we turn to the mathematical formalization of linguistic strengthening, first and foremost considering the design, commonly used in practical applications.

- “Very” – operation which narrows down the set and reduce the degree of the fuzzy element membership. Operation “very” can be defined as the square of the membership function

$$\mu_A^{very}(x) = (\mu_A(x))^2.$$

For instance, if the degree of membership of a tomato to the variety of “ripe” equals to 0.8, then the “very ripe” degree of membership equals to 0.64.

- “Extremely,” it serves the same purpose as “very”, but it does so to a greater extent, which can be done by raising  $\mu_A(x)$  to the third power:

$$\mu_A^{\text{extremely}}(x) = (\mu_A(x))^3.$$

In the previous example, the tomato will be “extremely ripe” if its degree of affiliation is 0.512.

- “Highly” just an extension of concentration.

$$\mu_A^{\text{highly}}(x) = (\mu_A^{\text{very}}(x))^2 = (\mu_A(x))^4.$$

- “More or less” – expand operation, which is a result of expanding the set and hence increasing the degree of membership of the fuzzy element. This operation can be represented as:

$$\mu_A^{\text{moreorless}}(x) = \sqrt{\mu_A(x)}.$$

Thus our tomato with a degree of 0.89 is “more or less ripe.”

- “Indeed” (similar with “in fact”, “truly”, “where it is so”, “perhaps”), the intensification of the operation which strengthens the meaning of a sentence. This can be done by increasing the degree of membership if it is less than 0.5 and if it is above 0.5 and reducing it if it is above 0.5, which can be written as:

$$\mu_A^{\text{indeed}}(x) = 2(\mu_A(x))^2 \text{ if } 0 \leq \mu_A(x) \leq 0.5$$

or






$$\mu_A^{\text{indeed}}(x) = 1 - 2(1 - \mu_A(x))^2 \text{ if } 0.5 < \mu_A(x) \leq 1.$$

Therefore, if the degree of membership of a tomato to a variety of “ripe” is equal to 0.8, then it would be “indeed ripe” with the degree of membership 0.92.

If the degree of membership of a tomato to a variety of “ripe” is equal to 0.2, then it would be “indeed ripe” with the degree of membership 0.08.

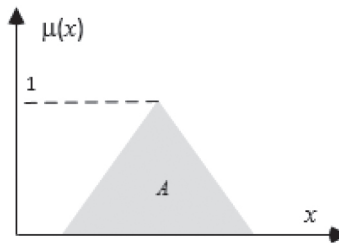
The Table 3.1 presents the linguistic hedges in fuzzy logic.

**Table 3.1**  
The presentation of linguistic hedges in fuzzy logic

Linguistic gain	Mathematical construction	Graphic illustration
Very	$(\mu_A(x))^2$	
Extremely	$(\mu_A(x))^3$	
Highly	$(\mu_A(x))^4$	
More or less	$\sqrt{\mu_A(x)}$	
Indeed	$2(\mu_A(x))^2$ if $0 \leq (\mu_A(x)) \leq 0.5$ $1-2(1-\mu_A(x))^2$ if $0.5 < (\mu_A(x)) \leq 1$	

### 3.4. Operations on fuzzy sets

In this section we confine ourselves to only those operations that will be needed further. Specifically these are: complement, inclusion (containment), intersection and union (for more see in Skalna et al. 2015, Zadeh 1996). We start with a set shown in Figure 3.8.

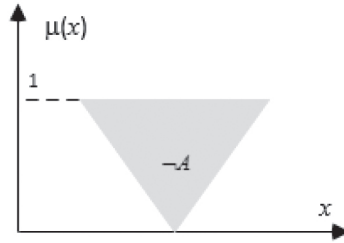


**Fig. 3.8.** The original set  $A$

#### Complement

The complement of a subset  $A$  is an opposite of this subset. For example, if we have a set of “ripe tomatoes”, its complement is a set “not ripe tomatoes”.

If  $A$  is a fuzzy set with membership function  $\mu_A(x)$ , the membership function of its complement  $\neg A$  is  $\mu_{\neg A}(x) = 1 - \mu_A(x)$  and is shown in Figure 3.9.



**Fig. 3.9.** The complement of the set  $A$

For example, if we have a fuzzy set of “light wind”, we can easily get the fuzzy set “not weaker than the wind”

Light wind =  $(1/0.3; 1/1.5; 0.71/4.0; 0.24/5.0; 0/5.5; 0/8.0; 0/10.8; 0/17.2; 0/32.7)$

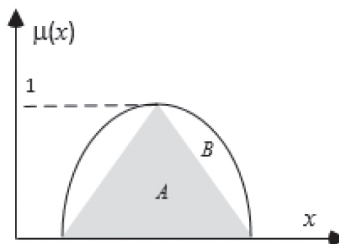
NOT weaker than the wind =  $(0/0.3; 0/1.5; 0.29/4.0; 0.76/5.0; 1/5.5; 1/8.0; 1/10.8; 1/17.2; 1/32.7)$

**Inclusion (containment)**

Like a kind of Babushka dolls in which each doll can contain many other dolls, one set can contain other sets. For example, a set “strong wind” contains the set “very strong wind”. Therefore, the “very strong wind” is a subset of the “strong wind”. In this case all crisp subset elements belong to the superset more fully and their membership values are 1. For the fuzzy sets, each cell may belong to the subset less than to the larger set. Elements of fuzzy subset may have a smaller membership degree in it than in a larger set as it is shown in Figure 3.10.

Strong wind =  $(0/10.8; 0.47/13.8; 1/2.17; 1/32.7)$

Very strong wind =  $(0/10.8; 0.22/13.8; 1/2.17; 1/32.7)$

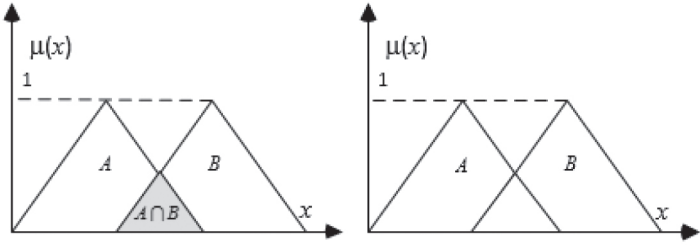


**Fig 3.10.** The inclusion  $A \subseteq B$



**Intersection**

In classical set theory, the intersection of two sets contains the elements that simultaneously belong to both sets. For fuzzy sets element may partly belong to both sets with different memberships. Thus, fuzzy intersection has lower membership in both sets for each element (see Fig. 3.11).



**Fig. 3.11.** The intersection of sets *A* and *B*

Fuzzy intersection operation for two fuzzy sets *A* and *B* on the universe *X* can be prepared as follows

$$\mu_{A \cap B}(x) = \min\{\mu_A(x), \mu_B(x)\} = \mu_A(x) \cap \mu_B(x) \text{ where } x \in X.$$

Let’s consider, for example, fuzzy sets “medium wind” and “strong wind”.

“Medium wind” = (0/3.4; 0.46/5.5; 1/8.0; 0.7/10.8; 0.37/13.8; 0/17.2; 0/32.7)

“Strong wind” = (0/3.4; 0/5.5; 0/8.0; 0/10.8; 0.47/13.8; 1/2.17; 1/32.7)

The intersection of these two sets can be written as follows

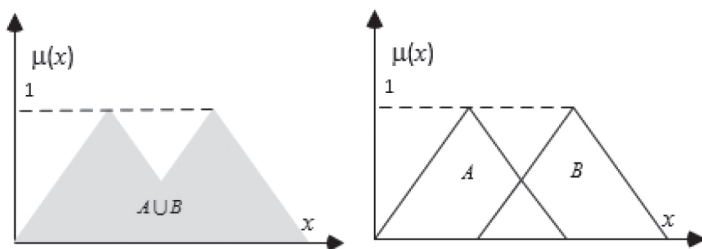
“Medium wind”  $\cap$  “Strong wind” = (0/3.4; 0/5.5; 0/8.0; 0/10.8; 0.37/13.8; 0/17.2; 0/32.7)

or

“Medium wind”  $\cap$  “Strong wind” = (0/10.8; 0.37/13.8; 0/17.2)

**Union**

In the case of crisp sets, the union of two sets comprises elements that fall into any of the sets. For fuzzy sets the union means an inverse of the intersection. That is, the union is the largest membership value of element in any set as it is presented in Figure 3.12.



**Fig. 3.12.** The union of sets  $A$  and  $B$

The fuzzy operation for forming the union of two fuzzy sets  $A$  and  $B$  on the universe  $X$  can be defined as

$$\mu_{A \cup B}(x) = \max\{\mu_A(x), \mu_B(x)\} = \mu_A(x) \cup \mu_B(x) \text{ where } x \in X.$$

Consider again the fuzzy sets “medium wind” and “strong wind”

$$\text{“Medium wind”} = (0/3.4; 0.46/5.5; 1/8.0; 0.7/10.8; 0.37/13.8; 0/2.17; 0/32.7)$$

$$\text{“Strong wind”} = (0/3.4; 0/5.5; 0/8.0; 0/10.8; 0.47/13.8; 1/2.17; 1/32.7)$$

Then

$$\text{“Medium wind”} \cup \text{“Strong wind”} = (0/3.4; 0.46/5.5; 1/8.0; 0.7/10.8; 0.47/13.8; 1/2.17; 1/32.7)$$

### 3.5. Properties of operations on fuzzy sets

There are some important properties of operations on fuzzy sets presented below.

#### Commutativity

$$A \cup B = B \cup A$$

$$A \cap B = B \cap A$$

Example:

$$\text{“Medium wind”} \cup \text{“Strong wind”} = \text{“Strong wind”} \cup \text{“Medium wind”}$$

$$\text{“Medium wind”} \cap \text{“Strong wind”} = \text{“Strong wind”} \cap \text{“Medium wind”}$$

### **Associativity**

$$A \cup (B \cap C) = (A \cup B) \cap C$$

$$A \cap (B \cup C) = (A \cap B) \cup C$$

Example:

$$\begin{aligned} \text{“Strong wind”} \cup (\text{“Light wind”} \cup \text{“Medium wind”}) &= \\ &= (\text{“Strong wind”} \cup \text{“Light wind”}) \cup \text{“Medium wind”} \end{aligned}$$

$$\begin{aligned} \text{“Strong wind”} \cap (\text{“Light wind”} \cap \text{“Medium wind”}) &= \\ &= (\text{“Strong wind”} \cap \text{“Light wind”}) \cap \text{“Medium wind”} \end{aligned}$$

### **Distributivity**

$$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$$

$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$$

Example:

$$\begin{aligned} \text{“Strong wind”} \cup (\text{“Light wind”} \cap \text{“Medium wind”}) &= \\ &= (\text{“Strong wind”} \cup \text{“Light wind”}) \cap (\text{“Strong wind”} \cup \text{“Medium wind”}) \end{aligned}$$

$$\begin{aligned} \text{“Strong wind”} \cap (\text{“Light wind”} \cup \text{“Medium wind”}) &= \\ &= (\text{“Strong wind”} \cap \text{“Light wind”}) \cup (\text{“Strong wind”} \cap \text{“Medium wind”}) \end{aligned}$$

**Additionally, we can note that**

$$A \cup A = A \text{ and } A \cap A = A$$

Example:

$$\text{“Medium wind”} \cup \text{“Medium wind”} = \text{“Medium wind”}$$

$$\text{“Medium wind”} \cap \text{“Medium wind”} = \text{“Medium wind”}$$

### **Identity**

$$A \cup \emptyset = A$$

$$A \cap \emptyset = \emptyset$$

$$A \cup X = X$$

$$A \cap X = A$$

Example:

$$\text{“Strong wind”} \cup \text{“No (Without) wind”} = \text{“Strong wind”}$$

$$\text{“Strong wind”} \cap \text{“No (Without) wind”} = \text{“No (Without) wind”}$$

$$\text{“Strong wind”} \cup \text{“Every wind”} = \text{“Every wind”}$$

$$\text{“Strong wind”} \cap \text{“Every wind”} = \text{“Strong wind”}$$

where “No (Without) wind” is an empty (null) set i.e. the set having the degree of membership of all 0 and “Every wind” is a set having all degrees of membership equal to 1.

### **Involution**

$$\neg(\neg A) = A$$

Example:

$$\text{NOT (NOT “Strong wind”)} = \text{“Strong wind”}$$

### **Transitivity**

If  $(A \subset B) \cap (B \subset C)$  then  $A \subset C$

Each set contains a subset of its subsets.

Example:

IF (“Extremely strong winds”  $\subset$  “Very strong wind”) and (“Very strong wind”  $\subset$  “Strong wind”) THEN (“Extremely strong winds”  $\subset$  “Strong wind”)

### De Morgan's laws

$$\neg(A \cap B) = \neg A \cup \neg B$$

$$\neg(A \cup B) = \neg A \cap \neg B$$

Example:

NOT (“Strong wind”  $\cap$  “Light wind”) = NOT “Strong wind”  $\cup$  NOT “Light wing”

NOT (“Strong wind”  $\cup$  “Light wind”) = NOT “Strong wind”  $\cap$  NOT “Light wind”

Using fuzzy set operations, their properties and linguistic strengthening, we can easily get a lot of fuzzy sets. For example, if we have a fuzzy set  $A$  as a “strong wind” and a fuzzy set  $B$  as a “light wind”, we can get a fuzzy set  $C$  as a “not very strong wind” and “not very light wind” or also a set  $D$  as “not very strong and very light wind” in result of the following operations:

$$\mu_C(x) = \left(1 - (\mu_A(x))^2\right) \cap \left(1 - (\mu_B(x))^2\right),$$

$$\mu_D(x) = \left(1 - (\mu_A(x))^4\right) \cap \left(1 - (\mu_B(x))^4\right).$$

## 3.6. Fuzzy inference rules

In 1973, Lotfi Zadeh published his second most influential paper (see Zadeh 1973, Zadeh 1975). This document describes a new approach to the analysis of complex systems. Not to mention that Zadeh’s publication outlines the application of assigning different values in fuzzy rules.

A fuzzy rule can be defined as the conditional expression in the following form

IF  $x$  is equal  $A$  THEN  $y$  is equal to  $B$ ,

where:

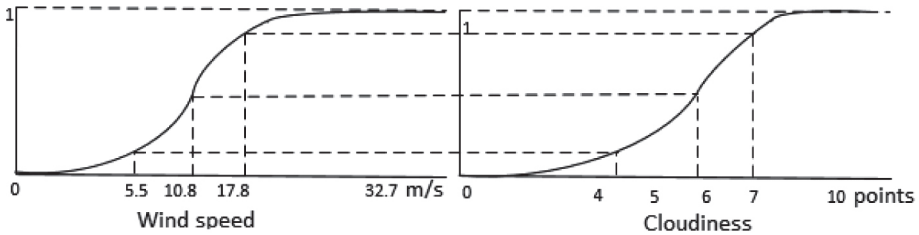
$x, y$  – linguistic variables,

$A, B$  – linguistic variables defined by fuzzy sets on the universe of  $X$  and  $Y$ , respectively.

In general, the argument IF-THEN consists of two separate parts: the evaluation one (IF part of the rule) and application one of the result (THEN part of the rule). The classic rule IF-THEN uses binary logic, that is, if the first part is true, then the second one is also true and vice versa. In fuzzy systems, to some extent, all of the rules work partially.

In the case of loyalty with some degree of IF conditions, we see that with the same degree it is true for the THEN condition.

For example, we can consider two fuzzy sets “Wind speed” and “Cloudiness” presented in Figure 3.13.



**Fig. 3.13.** The monotonic range of wind strength and cloudiness values (in points)

And what if the rule of fuzzy inference has several pieces?

For example:

IF the weather is frosty and snowy THEN there ski trip might be planned

IF the weather is good or pleasant THEN company picnic will succeed

The result of each rule is a fuzzy set, but usually we need to get a single number. In other words, we want to get the exact solution, not fuzzy. To get one crisp solution to the output variable, fuzzy expert system first aggregates all output fuzzy sets into a single output fuzzy set, and then collects the resulting fuzzy set in a single number.

The most frequently used fuzzy inference method is a so-called method of Mamdani.

In 1975, Professor, University of London, Ebrahim Mamdani built one of the first fuzzy systems to manage the combination of steam engines and boilers (see Mamdani, Assilian 1975, Mamdani 1977).

The process of fuzzy inference in Mamdani style is carried out in four stages: fuzzification of input variables, evaluation rules, the aggregation rules of inference and, finally, defuzzification, i.e. reduction to the definition.

### Fuzzy inference algorithms

Let  $x, y$  be the input variables that have clear values  $x_0, y_0$  and  $z$  – output membership of commutative function. Let's  $\mu_{A1} \cdot \mu_{A2} \cdot \mu_{B1} \cdot \mu_{B2} \cdot \mu_{C1} \cdot \mu_{C2}$  and rules be given

IF  $x$  is  $A1$  and  $y$  is  $B1$ , THEN  $z$  is  $C1$ ,  $A2$  is WHEN  $x$  and  $y$  is  $B2$ , THEN  $z$  is  $C2$

### Mamdani algorithm

In systems such as Mamdani's knowledge base is constructed from vague statements like "the  $z$  have alpha" with "and" tangles in "IF-THEN".

The stages of fuzzy inference are realized as follows:

1. Fuzzification: the degrees of truth are determined for prerequisites of each rule:  $\mu_{A1}(X_0)$ ,  $\mu_{A2}(X_0)$ ,  $\mu_{B1}(Y_0)$ ,  $\mu_{B2}(Y_0)$ . Fuzzification (introduction of fuzziness) is a fitting of the correspondence between the input numerical variable of the fuzzy inference system and the value of the membership function for the corresponding term of the linguistic variable.
2. Conclusion: It is determined the clipping level for prerequisites for each of the rules with using the operation of minimum

$$\alpha_1 = \min\{\mu_{A_1}(x_0), \mu_{B_1}(y_0)\}, \alpha_2 = \min\{\mu_{A_2}(x_0), \mu_{B_2}(y_0)\}.$$

Then there are the truncated membership functions

$$\mu_{C'_1}(z) = \min\{\alpha_1, \mu_{C_1}(x_0)\}, \mu_{C'_2}(z) = \min\{\alpha_2, \mu_{C_2}(x_0)\}.$$

3. Composition: a maximum operation allows to find truncated functions, which results in giving us a fuzzy subset of the final output for a variable membership function

$$\begin{aligned} \mu_{\Sigma}(z) = \mu_C(z) &= \max\{\mu_{C'_1}(z), \mu_{C'_2}(z)\} = \\ &= \max\{\min\{\alpha_1, \mu_{C_1}(z)\}, \min\{\alpha_2, \mu_{C_2}(z)\}\}. \end{aligned}$$

4. Reduction to the definition for  $z_0$  is produced by gravity

$$z_0 = \frac{\int_{Min}^{Max} z \cdot \mu_{\Sigma}(z) dz}{\int_{Min}^{Max} \mu_{\Sigma}(z) dz}.$$

Where *Min* and *Max* are the boundaries of the universe fuzzy variables. The algorithm is illustrated in Figure 3.14.

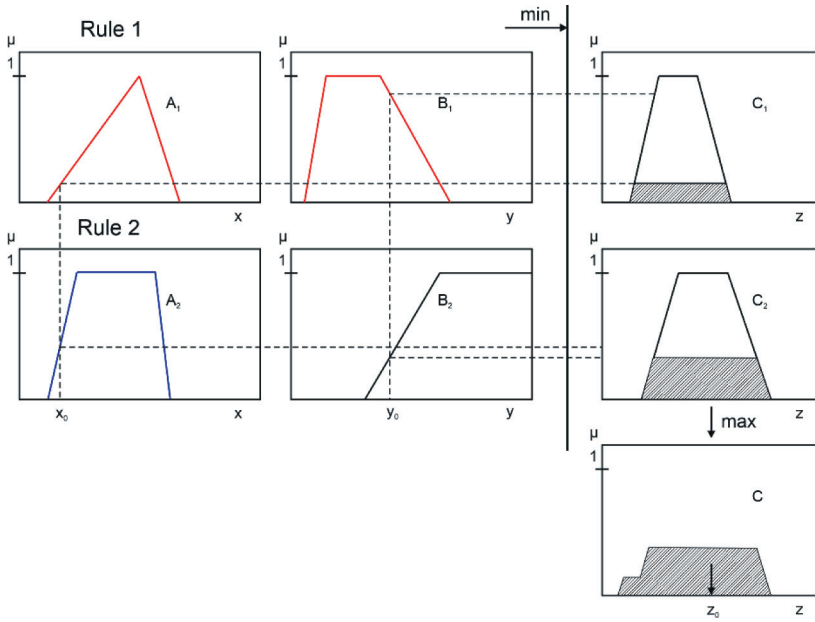


Fig. 3.14. The illustration of the Mamdani algorithm

### Tsukamoto algorithm

Baseline data and knowledge base are the same as in Mamdani algorithm, but additionally it is assumed that functions  $\mu_{C_1}(Z)$ ,  $\mu_{C_2}(Z)$  are monotonic.

Stages of fuzzy inference:

1. Fuzzification: there are degrees of truth for all prerequisites of each rule:  $\mu_{A_1}(X_0)$ ,  $\mu_{A_2}(X_0)$ ,  $\mu_{B_1}(Y_0)$ ,  $\mu_{B_2}(Y_0)$ .
2. Conclusion: There are the cut-off levels for each of the prerequisites for the operation of the rules using the minimum

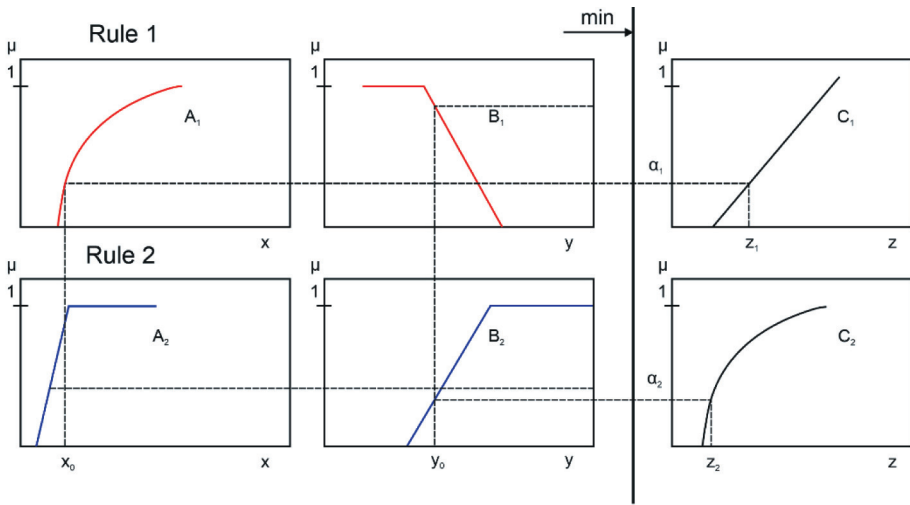
$$\alpha_1 = \min\{\mu_{A_1}(x_0), \mu_{B_1}(y_0)\}, \alpha_2 = \min\{\mu_{A_2}(x_0), \mu_{B_2}(y_0)\}.$$

3. Crisp values are then  $z_1$  and  $z_2$  in equations  $\alpha_1 = \mu_{C_1}(z)$ ,  $\alpha_2 = \mu_{C_2}(z)$ .
4. It clearly identifies the value of the output as a weighted average of  $z_1$  and  $z_2$

$$z_0 = \frac{\alpha_1 z_1 + \alpha_2 z_2}{\alpha_1 + \alpha_2}.$$

In general, the precise value  $z_0$  is defined by the centralization method. It can be illustrated in Figure 3.15.





**Fig. 3.15.** The illustration of the Tsukamoto algorithm

### Sugeno algorithm

In the algorithm of Sugeno the knowledge base is constructed from the rules in the following form:

IF  $x$  is  $A_1$  and  $y$  is  $B_1$ , THEN  $z_1 = a_1x + b_1y$ ;

IF  $x$  is  $A_2$  and  $y$  is  $B_2$ , THEN  $z_2 = a_2x + b_2y$ .

Stages of fuzzy inference (see in Sugeno 1985a, Sugeno 1985b, Takagi and Sugeno 1985, Terano et al. 1994):

1. Fuzzification: there are degrees of truth for all prerequisites of each rule prerequisites:  $\mu_{A_1}(X_0)$ ,  $\mu_{A_2}(X_0)$ ,  $\mu_{B_1}(Y_0)$ ,  $\mu_{B_2}(Y_0)$ .
2. Conclusion: there are the cut-off levels for each of the prerequisites for the operation of the rules using the minimum:

$$\alpha_1 = \min\{\mu_{A_1}(x_0), \mu_{B_1}(y_0)\}, \alpha_2 = \min\{\mu_{A_2}(x_0), \mu_{B_2}(y_0)\}.$$

3. Individual rights are the outputs:  $z_1^* = a_1x + b_1y$ ,  $z_2^* = a_2x + b_2y$  which clearly identifies the value of output:

$$z_0 = \frac{\alpha_1 z_1^* + \alpha_2 z_2^*}{\alpha_1 + \alpha_2}.$$

The algorithm is illustrated in Figure 3.16.

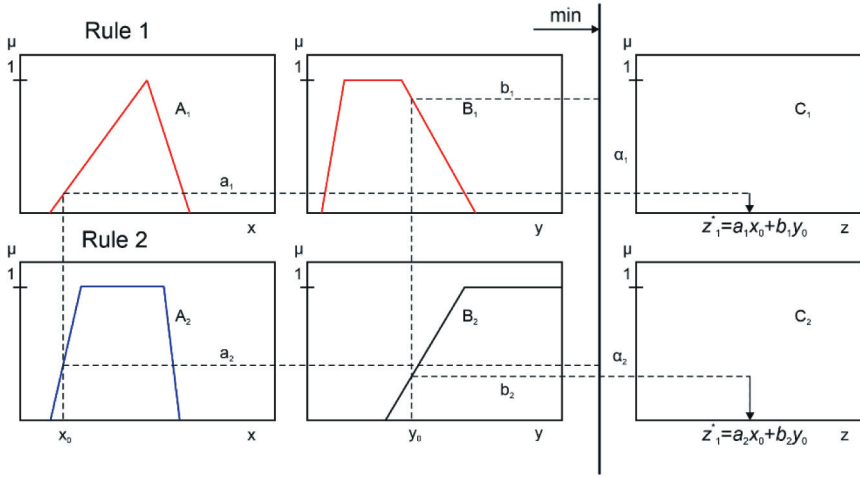


Fig. 3.16. The illustration of the Sugeno algorithm

### Larsen algorithm

The knowledge base in Larsen algorithm (see more in Larsen 1980) corresponds with the knowledge base for Mamdani algorithm.

1. Fuzzification: there are degrees of truth for each rule prerequisites:

$$\mu_{A_1}(X_0), \mu_{A_2}(X_0), \mu_{B_1}(Y_0), \mu_{B_2}(Y_0).$$

2. Conclusion: The cut-off levels are prerequisites for each of the operation rules for at least restricted to:

$$\alpha_1 = \min\{\mu_{A_1}(x_0), \mu_{B_1}(y_0)\},$$

$$\alpha_2 = \min\{\mu_{A_2}(x_0), \mu_{B_2}(y_0)\}.$$

In Larsen algorithm fuzzy subset of output variable for each rule is using the multiplication operator by the formula:

$$\mu_{C_1'}(z) = \alpha_1 \mu_{C_1}(z), \mu_{C_2'}(z) = \alpha_2 \mu_{C_2}(z).$$

3. Composition: a maximum operation is defined by association which finds private fuzzy subsets. It is the final fuzzy subset to exit with a variable membership function:

$$\mu_{\Sigma}(z) = \mu_C(z) = \max\{\mu_{C_1'}(z), \mu_{C_2'}(z)\} = \max\{\alpha_1 \mu_{C_1}(z), \alpha_2 \mu_{C_2}(z)\}.$$

4. Reduction to the definition is also produced by the method of gravity center.

The Larsen algorithm is illustrated in Figure 3.17.

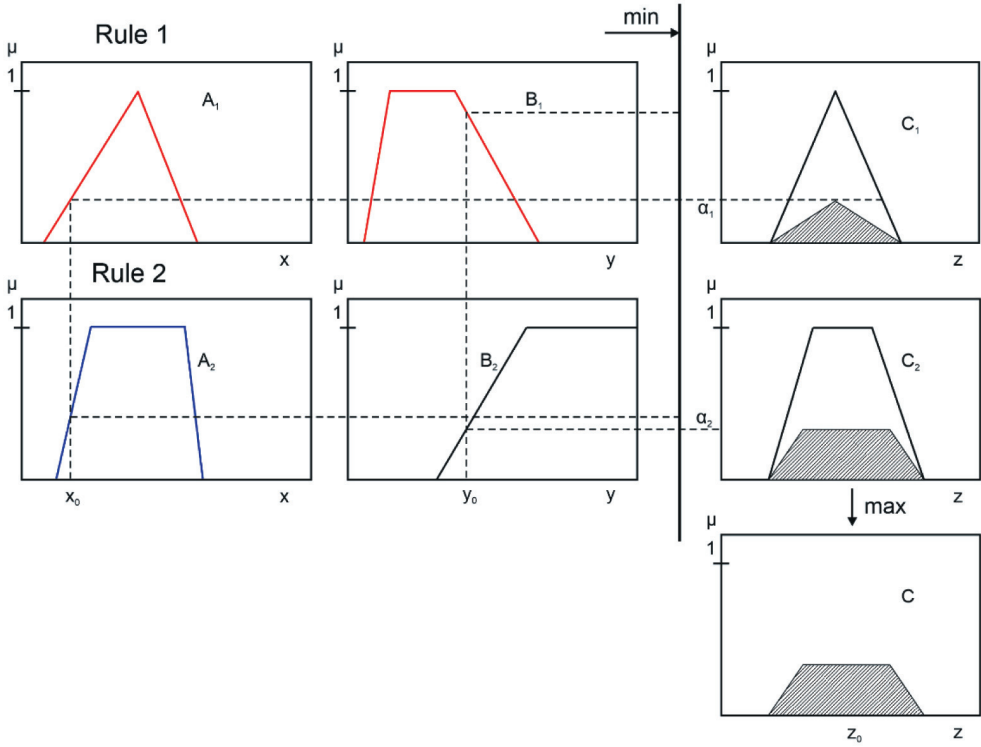


Fig. 3.17. The illustration of the Larsen algorithm

Let's take a look at an example. It is required to assess the level of risk associated with software development projects. We assume that there is a possibility of varying only two parameters – the volume of funding and project staffing size (see in Chiu and Park 1994, Shin and Wang 2010).

**Step 1**

At the first stage, we transform input crisp to input fuzzy variables. As we have two input parameters, we can associate it with two distinct values, respectively. The first value is the size of a full-time project schedule. The second value is the amount of funding for the project.

Suppose that we have the following resources – project financing = 35% and project stylesheet = 60%. We can get fuzzy values for these precise values using the membership function of the corresponding sets. Sets defined for project financing are 'insufficient', 'critical' and 'adequate'. Sets defined for project stylesheet are 'small' and 'large'.

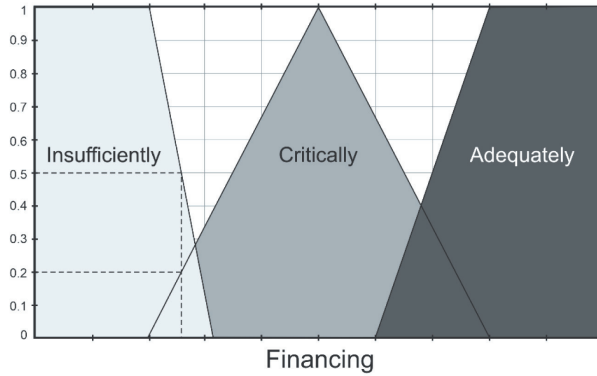
Thus, we have the following values for the fuzzy project financing:

$$\mu_{\text{financing}} \text{ 'insufficient' } (35) = 0.5,$$

$$\mu_{\text{financing}} \text{ 'critical' } (35) = 0.2,$$

$$\mu_{\text{financing}} \text{ 'adequately' } (35) = 0.0.$$

We have the following visual representation of this procedure (see Fig. 3.18).



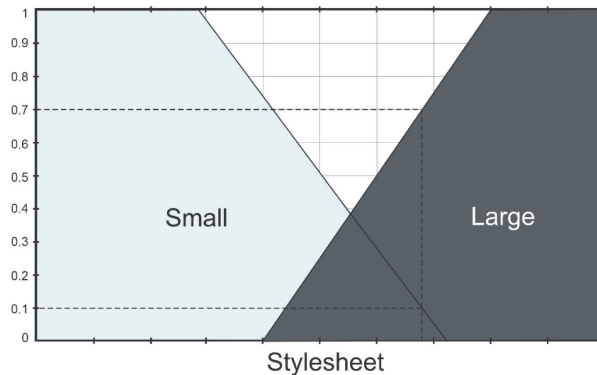
**Fig. 3.18.** The fuzzy representation of the project financing

Below there are values for the fuzzy project sheet:

$$\mu_{\text{stylesheet}} \text{ 'small' } (60) = 0.1,$$

$$\mu_{\text{stylesheet}} \text{ 'high' } (60) = 0.7.$$

On the Figure 3.19, below, there is a visual representation of the procedure.



**Fig. 3.19.** The fuzzy representation of the project stylesheet state

## Rules

Now that there are fuzzy values, you can use fuzzy rules to obtain the final fuzzy values. Rules are as follows:

1. IF project financing is adequate or project stylesheet is small, THEN the risk is law.
2. IF project financing is critical, and project stylesheet is large, THEN the risk is normal.
3. IF project financing is insufficient THEN the risk is high.

**Rule 1** – IF project financing is adequate or project stylesheet is small, THEN the risk is law.

Rules which are concerning the disjunction, OR, are estimated by using the UNION operator:

$$\mu_{A \cup B}(x) = \max\{\mu_A(x), \mu_B(x)\},$$

$$\begin{aligned}\mu_{\text{risk}} \text{ 'low'} &= \max\{\mu_{\text{financing}} \text{ 'adequate'} (35) = 0.0; \mu_{\text{stylesheet}} \text{ 'small'} (60) = 0.1\} = \\ &= \max\{0.0; 0.1\} = 0.1.\end{aligned}$$

And an alternative method for calculating the disjunction i.e. through the algebraic sum can take the following form:

$$\mu_{A \cup B}(x) = \mu_A(x) + \mu_B(x) - \mu_A(x) \cdot \mu_B(x),$$

$$\mu_{\text{risk}} \text{ 'low'} = 0.0 + 0.1 - 0.1 \cdot 0.0 = 0.1.$$

**Rule 2** – IF project financing is critical, and project stylesheet is large, THEN the risk is normal.

In conjunction fuzzy rules are calculated as follows:

$$\mu_{A \cap B}(x) = \min\{\mu_A(x), \mu_B(x)\},$$

$$\begin{aligned}\mu_{\text{risk}} \text{ 'normal'} &= \min\{\mu_{\text{financing}} \text{ 'critical'} (35) = 0.2, \mu_{\text{stylesheet}} \text{ 'large'}(60) = 0.7\} = \\ &= \min\{0.2, 0.7\} = 0.2.\end{aligned}$$

Alternatively, the same rule can be evaluated by using the multiplication:

$$\mu_{A \cap B}(x) = \mu_A(x) \cdot \mu_B(x),$$

$$\mu_{\text{risk}} \text{ 'normal'} = 0.2 \cdot 0.7 = 0.14.$$

**Rule 3** – IF project financing is insufficient, THEN the risk is high.

$$\mu_{\text{risk}} \text{ 'high'} = 0.2 \cdot 0.7 = 0.14.$$

### The results of valuation rules

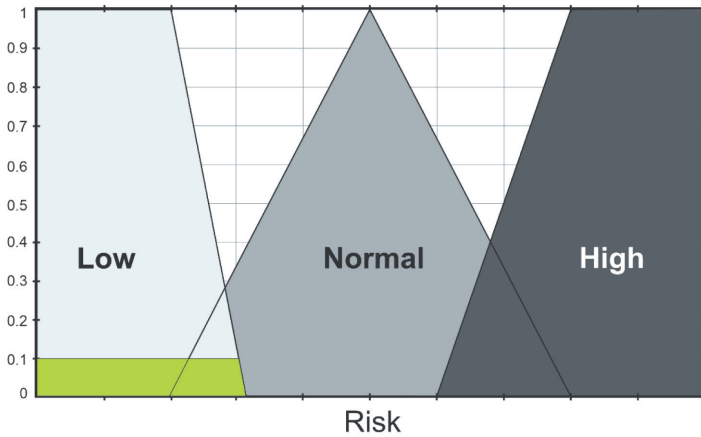
The result of evaluation rules can be shown as follows:

$\mu_{\text{risk}} \text{ 'low' } (z) = 0.1$  (Fig. 3.20),

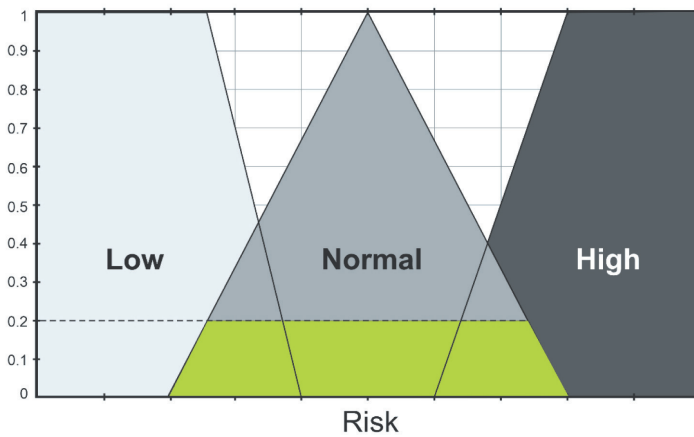
$\mu_{\text{risk}} \text{ 'normal' } (z) = 0.2$  (Fig. 3.21),

$\mu_{\text{risk}} \text{ 'high' } (z) = 0.5$  (Fig. 3.22).

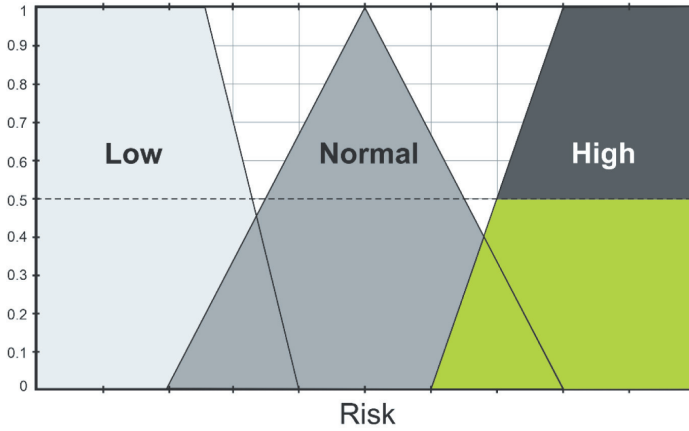
Now we use the results to scale the membership functions (see Fig. 3.20–3.23).



**Fig. 3.20.** The evaluation of the low risk project

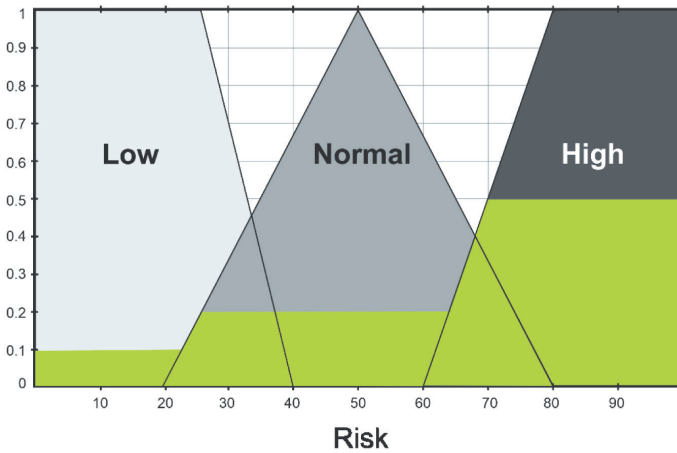


**Fig. 3.21.** The evaluation of the normal risk project



**Fig. 3.22.** The evaluation of high-risk project

We carry out the union of the whole scalability to produce the final result. The result is again displayed in green color in Figure 3.23.



**Fig. 3.23.** The assessment of the overall project risk

### 3.7. Defuzzification

Defuzzification can be performed in several different ways (see in Skalna et al. 2015, Zadeh 1996). The most popular method is the centroid method.

### **Centroid method**

This method applies in calculating the center of gravity for the area under the curve.

$$\text{COG} = \frac{\sum_{x=a}^b \mu_A(x)x}{\sum_{x=a}^b \mu_A(x)}.$$

### **Bisector**

There is a vertical line that divides the region into two subregions of equal area. The bisector sometimes, but not always, coincides with the centroid line.

### **Average value**

It is assumed that the plateau at the maximum value function takes a finite average value.

### **The lowest value of the maximum**

It is assumed that the plateau at the maximum ultimate function takes the smallest of the values, which it covers.

### **The highest value of the maximum**

It is assumed that the plateau at the maximum ultimate function takes the largest value which it covers.

Referring to the example given above we can calculate the final value of our project risk.

We use centroid method to find the final value of the fuzzy risks associated with our project.

$$\text{COG} = \frac{(0+10+20) \cdot 0.1 + (30+40+50+60) \cdot 0.2 + (70+80+90+100) \cdot 0.5}{0.1 \cdot 3 + 0.2 \cdot 4 + 0.5 \cdot 4} = 67.4.$$

As a result, taking the definitions of this project risk into account it is 67.4%.

Notice, that application of the Sugeno algorithm gives a slightly different risk value i.e. 65%.



## Ranging

Note that quite often analyses needs to streamline the existing set of fuzzy sets, in order to rank them. For example, it is required to assign each of them having a rating for a set of projects (see in Chernov et al. 2015, Shin, Wang 2010). As in previous cases, we use the estimation in the form of trapezoidal and triangular fuzzy numbers, as in a special case.

So, after completing each project, we have receive an overall assessment of a number of trapezoidal numbers. Afterwards we can organize projects in accordance with the assigned values.

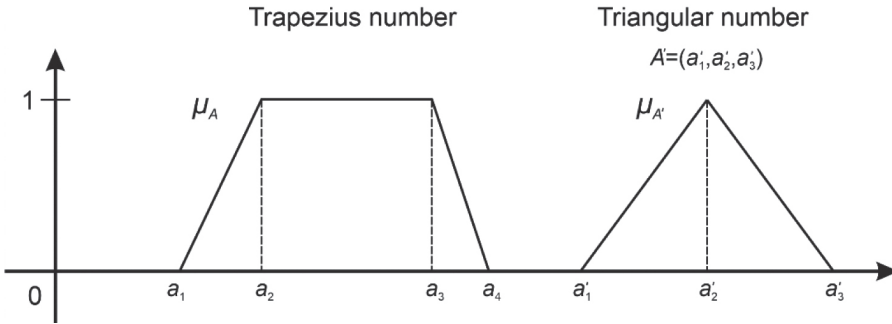


Fig. 3.24. Trapezoidal and triangular fuzzy numbers

For comparison of fuzzy numbers, there are several different methods:

- 1) The method by Chew Park. Let's fix parameter  $w$ . Every trapezoidal number  $A = (a_1, a_2, a_3, a_4)$  is associated with the crisp number

$$cp(A) = \frac{1}{4}(a_1 + a_2 + a_3 + a_4) + \frac{w}{2}(a_2 + a_3).$$

The ordering is performed by ascending value  $cp(A)$ .

- 2) The Chang method. Let's assume the trapezoidal number  $A = (a_1, a_2, a_3, a_4)$  keeps values in increasing order. The ordering is performed by ascending value  $ch(A)$

$$ch(A) = \frac{1}{6}(a_3^2 + a_3a_4 + a_4^2 - a_1^2 - a_1a_2 - a_2^2).$$

- 3) The method by Kaufman–Gupta. For the trapezoidal number  $A$  three values are calculated as follows

$$kg_1(A) = \frac{1}{6}(a_1 + 2a_2 + 2a_3 + a_4), \quad kg_2(A) = \frac{1}{6}(a_3 + a_4), \quad kg_3(A) = a_4 - a_1.$$

We assume that  $A \geq B$ , if

$kg_1(A) > kg_1(B)$ , or

$kg_1(A) = kg_1(B)$  and  $kg_2(A) > kg_2(B)$ , or

$kg_1(A) = kg_1(B)$  and  $kg_2(A) = kg_2(B)$  and  $kg_3(A) > kg_3(B)$ .

- 4) The Jane's method. The method specifies the procedure for a set of fuzzy numbers  $A_1, A_2, \dots, A_n$ . Let these numbers lie in the interval from  $b_1$  to  $b_2$ . Then fuzzy number  $B = (b_1, b_2, \infty, \infty)$  can be considered as "large number". For each  $A_i$ , one may consider extending its meaning so that  $A_i$  is a "big".

$$\text{Pos}(A_i \in B) = \max_x \min(\mu_{A_i}(x), \mu_B(x)).$$

Set  $A_1, A_2, \dots, A_n$  is ordered by ascending values  $\text{Pos}(A_i \in B)$ .

- 5) The method of Dubois-Prada. Like the previous method, there is a set of fuzzy numbers  $A_1, A_2, \dots, A_n$ . Each number  $A_i$  meets its degree of dominance over the rest of the numbers:

$$\text{PD}(A_i) = \text{Pos} \left( A_i \geq \max_{j \neq i} A_j \right) = \min_{j \neq i} \max_{x,y} \min \{ \mu_{A_i}(x), \mu_{A_j}(y) \}.$$

The numbers are ordered by ascending values of  $\text{PD}(A_i)$ .

Considering the comparison methods in general, they may give different results.

### Example

Suppose that we have three projects with the estimated trapezoidal number as follows

$$A_1 = (3, 5, 5, 9), \quad A_2 = (3, 7, 7, 8), \quad A_3 = (1, 6, 6, 10).$$

By the method of Chew Park with the parameter  $w = 1$ , we take the following order:

$$cp(A_1) = 10.5 < cp(A_3) = 11.75 < cp(A_2) = 13.25.$$

The best is the second project ( $A_2$ ), followed by the third ( $A_3$ ) and the first ( $A_1$ ) ones. Chang method leads to the following result

$$\text{ch}(A_2) = 15 < \text{ch}(A_1) = 17 < \text{ch}(A_3) = 25.5,$$

that is, the second project ( $A_2$ ) is the worst.

By the method of Kaufman–Gupta it is obtained

$$\text{kg}_1(A_1) = 5.33 < \text{kg}_1(A_3) = 5.83 < \text{kg}_1(A_2) = 6.5,$$

which coincides with the result of the method of Park Chew.

In the method of Jain it is defined the plurality of large numbers as  $B = (0, 10, \infty, \infty)$ . Then

$$\text{Pos}(A_1 \in B) = 6.43 < \text{Pos}(A_3 \in B) = 7.14 < \text{Pos}(A_2 \in B) = 7.27.$$

The order coincides with the order of Chew Park.

Application of the method of Dubois-Prada gives the following inequality

$$PD(A_1) = 0.75 < PD(A_3) = 0.875 < PD(A_2) = 1.$$

This leads to the following list “Project 1 < Project 3 < Project 2”.

### 3.8. The choice of alternatives using fuzzy inference rules

Consider a multi-criteria selection methods based on complex descriptions of alternatives aggregation rules with information as fuzzy sentences, such as in making decisions (see for instance in Abdullah 2013, Baldwin, Xie 2005, *Building Classification Models: ID3...*, Efstahiou, Rajkovic 1980, Ekel 2002, Evans, Lohse 2011, Federizzi, Pasi 2008, Larsen 1980, Lee 1990, Skalna et al. 2015).

Let  $U$  be a set of elements,  $A$  its fuzzy subset,  $\mu$  degree of element memberships as the number from the unit interval  $[0, 1]$ . Subset  $A$  has a linguistic variable values  $H$ .

Let the solution set be characterized by a set of criteria  $X_1, X_2, \dots, X_p$  i.e. linguistic variables on the base sets  $U_1, U_2, \dots, U_p$  respectively. A set of several criteria with the corresponding values characterizes the presentation of satisfactory (acceptability) solutions. Variable  $S$  «satisfactory» is also linguistic. Example statements can be written as follows

$$d_1 : \text{“IF } X_1 = \text{LOW and } X_2 = \text{GOOD, THEN } S = \text{HIGH”}.$$

In general, a statement has the form

$$d_i : \text{“IF } X_1 = A_{1,i} \text{ and } X_2 = A_{2,i} \text{ and... and } X_p = A_{p,i} \text{ THEN } S = B_i”.$$

The intersection  $(X_1=A_{1,i} \cap X_2=A_{2,i} \cap \dots \cap X_p = A_{p,i})$  is across  $X = A_i$ . The operation of fuzzy sets intersection corresponds to finding the minimum of the membership functions

$$\mu_{A_i}(v) = \min_{v \in V} (\mu_{A_{i1}}(u_1), \mu_{A_{i2}}(u_2), \dots, \mu_{A_{ip}}(u_p)).$$

Where  $V = U_1 \times U_2 \times \dots \times U_p$ ;  $v = (u_1, u_2, \dots, u_p)$ ;  $\mu_{A_{i,j}}(u_j)$  – value of the element supplies  $u_j$  fuzzy set  $A_{i,j}$ . Then the rule will take the form

$$d_i : \text{"IF } X = A_i \text{ THEN } S = B_i\text{"}$$

We denote the base set ( $U$  or  $V$ ) by  $W$ . Then  $A_i$  is a fuzzy subset of  $W$ , while  $B_i$  a fuzzy subset of the unit interval  $I = [0, 1]$ .

The implication ("IF... THEN... ") of fuzzy sets is expressed as follows

$$\mu_H(w, i) = \min_{w \in W} (1, (1 - \mu_A(w) + \mu_B(i))),$$

where  $H$  is a fuzzy subset on  $W \times I$ ,  $w \in W$ ,  $i \in I$ . Similarly expressions  $(d_1, d_2, \dots, d_q)$  are converted into a set of  $(H_1, H_2, \dots, H_q)$ . Their union is a set  $D$  in the following form

$$D = H_1 \cap H_2 \cap \dots \cap H_q$$

and for each  $(w, i) \in W \times I$  it can be written in the following form

$$\mu_D(w, i) = \min_{w \in W} \{\mu_{H_j}(w, i)\}, j = 1, 2, \dots, q.$$

Consider the choice of alternatives, each of which is described by a fuzzy subset  $C$  of  $W$ . Satisfactory alternative is based on a composite output rules

$$G = C \circ D,$$

where  $G$  is a fuzzy subset of the interval  $I$ . Then

$$\mu_G(i) = \max_{w \in W} \{\min\{\mu_C(w), \mu_D(w, i)\}\}.$$

A comparison of the alternatives occurs on the basis of point estimates. For fuzzy set  $A \subset I$  it is define  $\alpha$ -level as follows ( $\alpha \in [0, 1]$ )

$$A_\alpha = \{x \mid \mu_A(x) \geq \alpha, x \in I\}.$$

For each  $A_\alpha$  we can calculate the average number of elements  $M(A_\alpha)$ :

1. For a set of  $n$  elements  $M(A_\alpha) = \frac{1}{n} \sum \{x_i \mid x_i \in A_\alpha\}$ ,
2. For  $\{A_\alpha = a \leq x \leq b\}$ ,  $M(A_\alpha) = \frac{a+b}{2}$ ,
3. For  $0 \leq a_1 \leq b_1 \leq a_2 \leq b_2 \leq \dots \leq a_n \leq b_n \leq 1$ ,

$$A_\alpha \bigcup_{i=1}^n \{a_i \leq x \leq b_i\} \text{ we get } M(A_\alpha) = \frac{\sum_{i=1}^n \frac{a_i + b_i}{2} (b_i - a_i)}{\sum_{i=1}^n (b_i - a_i)}.$$

Then the expected value of  $A$  can be calculated as follows

$$F(A) = \frac{1}{\alpha_{\max}} \int_0^{\alpha_{\max}} M(A_\alpha) d\alpha,$$

where  $\alpha_{\max}$  is the value at which  $A$  has a maximum.

When choosing an option for each of them, a satisfactory and calculated appropriate point estimate is considered, regarded the best alternative with the highest value.

### **Task – the choice of the most suitable candidate from the list**

This example is based on the work by Borisov et al. (Borisov et al. 1990). Suppose that we have the following result of the discussion over applicants:

$d_1$ : “IF a candidate is an experienced researcher who has a certain production experience and teaching experience in high school, THEN he is the adequate”,

$d_2$ : “IF he has to  $d_1$  and can teach the theory of information systems, THEN he is more than satisfactory”,

$d_3$ : “IF he has to  $d_2$  and he is a person who can find customer for high-tech products, THEN he is faultless”,

$d_4$ : “IF all agreed that he has  $d_3$  but he has not the teaching ability of the theory of information systems, THEN he is very satisfactory”,

$d_5$ : “IF a candidate is a very experienced researcher who has the ability to find a customer and is a good teacher, but no manufacturing experience, THEN he is satisfactory”,

$d_6$ : “IF a candidate is an investigator who has no qualifications or ability to teach, THEN he is unsatisfying”.

Analysis of the data frame gives five criteria used in decision-making:

- $X_1$  – research capacity,
- $X_2$  – manufacturing experience,
- $X_3$  – teaching experience,
- $X_4$  – teaching experience of information systems theory,
- $X_5$  – ability to find customers.

We will measure these variables at baseline set  $U$  candidates.

$d_1$  “IF  $X_1 = X_2 =$  educated and some experience and  $X_3 =$  good, THEN  $Y =$  satisfied”,

$d_2$  “IF  $X_1 = X_2 =$  educated and some experience, and  $X_3 = X_4 =$  good abilities, THEN  $Y =$  more than satisfactory”,

$d_3$  “IF  $X_1 = X_2 =$  educated and some experience, and  $X_3 = X_4 =$  good abilities and  $X_5 =$  capable, THEN  $Y =$  perfect”,

$d_4$  “IF  $X_1 = X_2 =$  educated and some experience and  $X_3 = X_5 =$  good abilities, THEN  $Y =$  very satisfying”,

$d_5$  “IF  $X_1 =$  very educated and  $X_2 =$  no experience, and  $X_3 = X_5 =$  good abilities, THEN  $Y =$  satisfactory”,

$d_6$  “IF  $X_1 =$  not educated and  $X_3 =$  not capable of teaching, THEN  $Y =$  not satisfactory”.

The variable  $Y$  defined on the set  $J = \{0; 0.1; 0.2; \dots; 0.9; 1\}$ :

- satisfactory is defined as  $\mu_S(x) = x$  for  $x \in J$ ,
- more than satisfactory is expressed by  $\mu_{MS}(x) = \sqrt[3]{x}$  for  $x \in J$ ,
- perfect is described as follows,

$$\mu_P(x) = \begin{cases} 1 & \text{if } x = 1 \\ 0 & \text{if } x \neq 1, \end{cases}$$

- very satisfying is defined as  $\mu_{VS}(x) = x^2$  for  $x \in J$ ,
- not satisfactory  $\mu_{US}(x) = 1 - x$  for  $x \in J$ .

Elections are held five candidates  $U = \{u_1, u_2, u_3, u_4, u_5\}$ . We have the following assessment of each candidate

$$\begin{aligned} A = \text{EDUCATED EXPLORER} &= \\ &= \{0.8/u_1, 0.6/u_2, 0.5/u_3, 0.1/u_4, 0.3/u_5\} \end{aligned}$$

$$B = \text{SOME PRODUCTION EXPERIENCE} = \\ = \{0.5/u_1, 1/u_2, 0/u_3, 0.5/u_4, 1/u_5\}$$

$$G = \text{GOOD CAPACITY IN TEACH} = \\ = \{0.6/u_1, 0.9/u_2, 1/u_3, 0.7/u_4, 1/u_5\}$$

$$D = \text{CAPABILITY TEACHING INFORMATION SYSTEMS} = \\ = \{1/u_1, 0.3/u_2, 1/u_3, 0/u_4, 0/u_5\}$$

$$E = \text{CAPABILITY REQUEST EXTERNAL FINANCING} = \\ = \{0/u_1, 0.5/u_2, 1/u_3, 0.8/u_4, 0.1/u_5\}$$

Afterwards it can take the form of knowledge fragments:

$$d_1 \text{ IF } X = A \text{ and } B \text{ and } C, \text{ THEN } Y = S$$

$$d_2 \text{ IF } X = A \text{ and } B \text{ and } C \text{ and } D, \text{ THEN } Y = MS$$

$$d_3 \text{ IF } X = A \text{ and } B \text{ and } C \text{ and } D \text{ and } E, \text{ THEN } Y = P$$

$$d_4 \text{ IF } X = A \text{ and } B \text{ and } C \text{ and } E, \text{ THEN } Y = VS$$

$$d_5 \text{ IF } X = \text{very } A \text{ and not } B \text{ and } C \text{ and } E, \text{ THEN } Y = S$$

$$d_6 \text{ IF } X = \text{not } A \text{ or not } C, \text{ THEN } Y = US$$

Then, given that the operation of intersection of fuzzy sets corresponds to finding the minimum of their membership functions and the operation of union corresponds to their maximum, and it should be used the  $\cup$  operations, we get the following sets:

$$d_1: \mu_{M_1}(u) = \min\{\mu_A(u), \mu_B(u), \mu_C(u)\} \\ M_1 = \{0.5/u_1, 0.6/u_2, 0/u_3, 0.1/u_4, 0.3/u_5\}$$

$$d_2: \mu_{M_2}(u) = \min\{\mu_A(u), \mu_B(u), \mu_C(u), \mu_D(u)\} \\ M_2 = \{0.5/u_1, 0.3/u_2, 0/u_3, 0/u_4, 0/u_5\}$$

$$d_3: \mu_{M_3}(u) = \min\{\mu_A(u), \mu_B(u), \mu_C(u), \mu_D(u), \mu_E(u)\} \\ M_3 = \{0/u_1, 0.3/u_2, 0/u_3, 0/u_4, 0/u_5\}$$

$$d_4: \mu_{M_4}(u) = \min\{\mu_A(u), \mu_B(u), \mu_C(u), \mu_E(u)\}$$

$$M_4 = \{0/u_1, 0.5/u_2, 0/u_3, 0.1/u_4, 0.1/u_5\}$$

$$d_5: \mu_{M_5}(u) = \min\{\mu_{A^2}(u), 1 - \mu_B(u), \mu_C(u), \mu_E(u)\}$$

$$M_5 = \{0/u_1, 0/u_2, 0.25/u_3, 0.01/u_4, 0/u_5\}$$

$$d_6: \mu_{M_6}(u) = \max\{1 - \mu_A(u), 1 - \mu_C(u)\}$$

$$M_6 = \{0.4/u_1, 0.4/u_2, 0.5/u_3, 0.9/u_4, 0.7/u_5\}$$

In this way we get rules:

$$d_1: \text{If } X = M_1, \text{ then } Y = S$$

$$d_2: \text{If } X = M_2, \text{ then } Y = MS$$

$$d_3: \text{If } X = M_3, \text{ then } Y = P$$

$$d_4: \text{If } X = M_4, \text{ then } Y = VS$$

$$d_5: \text{If } X = M_5, \text{ then } Y = S$$

$$d_6: \text{If } X = M_6, \text{ then } Y = US$$

Using implication and its conversions “IF  $X = F$ , THEN  $Y = Q$ ” in the expression  $\mu_D(u, i) = \min(1, 1 - \mu_M(u) + \mu_Q(u))$  for each pair  $(u, i) \in U \times J$  we obtain the following fuzzy subsets of  $U \times J$ .

	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
$u_1$	0.5	0.6	0.7	0.8	0.9	1	1	1	1	1	1
$u_2$	0.4	0.5	0.6	0.7	0.8	0.9	1	1	1	1	1
$u_3$	1	1	1	1	1	1	1	1	1	1	1
$u_4$	0.9	1	1	1	1	1	1	1	1	1	1
$u_5$	0.7	0.8	0.9	1	1	1	1	1	1	1	1



	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
$D_2 =$	$u_1$	0.5	0.53	0.59	0.66	0.75	0.85	0.96	1	1	1
	$u_2$	0.7	0.73	0.79	0.86	0.95	1	1	1	1	1
	$u_3$	1	1	1	1	1	1	1	1	1	1
	$u_4$	1	1	1	1	1	1	1	1	1	1
	$u_5$	1	1	1	1	1	1	1	1	1	1

	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
$D_3 =$	$u_1$	1	1	1	1	1	1	1	1	1	1
	$u_2$	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	1
	$u_3$	1	1	1	1	1	1	1	1	1	1
	$u_4$	1	1	1	1	1	1	1	1	1	1
	$u_5$	1	1	1	1	1	1	1	1	1	1

	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
$D_4 =$	$u_1$	1	1	1	1	1	1	1	1	1	1
	$u_2$	0.5	0.51	0.54	0.59	0.66	0.75	0.86	0.99	1	1
	$u_3$	1	1	1	1	1	1	1	1	1	1
	$u_4$	0.9	0.91	0.94	0.99	1	1	1	1	1	1
	$u_5$	0.9	0.91	0.94	0.99	1	1	1	1	1	1

	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
$D_5 =$	$u_1$	1	1	1	1	1	1	1	1	1	1
	$u_2$	1	1	1	1	1	1	1	1	1	1
	$u_3$	0.75	0.85	0.95	1	1	1	1	1	1	1
	$u_4$	0.99	1	1	1	1	1	1	1	1	1
	$u_5$	1	1	1	1	1	1	1	1	1	1

	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
$D_6 =$	$u_1$	1	1	1	1	1	1	0.9	0.8	0.7	0.6
	$u_2$	1	1	1	1	1	1	0.9	0.8	0.7	0.6
	$u_3$	1	1	1	1	1	0.9	0.8	0.7	0.6	0.5
	$u_4$	1	1	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2
	$u_5$	1	1	1	1	0.9	0.8	0.7	0.6	0.5	0.4

The result is an overall functional solution calculated as follows:

$$D = D_1 \cap D_2 \cap D_3 \cap D_4 \cap D_5 \cap D_6 \text{ i.e. } \mu_D(u, i) = \min_{j=1, \dots, 6} \{\mu_{D_j}(u, i)\},$$

	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
$u_1$	0.5	0.53	0.59	0.66	0.75	0.85	0.96	0.9	0.8	0.7	0.6
$u_2$	0.4	0.5	0.54	0.59	0.66	0.7	0.7	0.7	0.7	0.7	0.6
$u_3$	0.75	0.85	0.95	1	1	1	0.9	0.8	0.7	0.6	0.5
$u_4$	0.9	0.91	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1
$u_5$	0.7	0.8	0.9	0.99	0.9	0.8	0.7	0.6	0.5	0.4	0.3

To calculate each of satisfactory alternatives for applicable rule, output fuzzy composite medium can be  $E_k = G_k \circ D$  where  $E_k$  is the degree of satisfaction of alternative  $k$ ,  $G_k$  is a map alternative  $k$  in the form of a fuzzy subset of  $U$ , and  $D$  is a functional solution. Then

$$\mu_{E_k}(i) = \max_{u \in U} \left( \min(\mu_{G_k}(u), \mu_{D_k}(u, i)) \right).$$

Furthermore, in this case  $\mu_{G_k}(u) = 0, u \neq u_k; \mu_{G_k}(u) = 1, u = u_k$  here  $\mu_{E_k}(i) = \mu_D(u_k, i)$ . In other words,  $E_k$  is the  $k$ -th row of the matrix  $D$ .

Returning to the example now we apply a procedure for comparison of fuzzy sets  $E_1, E_2, E_3, E_4, E_5$  in the unit interval for the best solutions.

For the first alternative

$$E_1 = \{0.5/0; 0.53/0.1; 0.59/0.2; 0.66/0.3; 0.75/0.4; 0.85/0.5; 0.96/0.6; 0.9/0.7; 0.8/0.8; 0.7/0.9; 0.6/1\}.$$

We compute levels sets  $E_{j, \alpha}$ . Their power  $M(E_{j, \alpha})$  is equal  $M(E_{j, \alpha}) = \frac{1}{n} \sum_{i=1}^n x_i$ .

Where:

$$0 \leq \alpha \leq 0.5; d\alpha = 0.5;$$

$$E_{1, \alpha} = \{0; 0.1; 0.2; 0.3; 0.4; 0.5; 0.6; 0.7; 0.8; 0.9; 1\}; M(E_{1, \alpha}) = 0.5;$$

$$0.5 \leq \alpha \leq 0.53; d\alpha = 0.03;$$

$$E_{1,\alpha} = \{0.1; 0.2; 0.3; 0.4; 0.5; 0.6; 0.7; 0.8; 0.9; 1\}; M(E_{1,\alpha}) = 0.55;$$

$$0.53 \leq \alpha \leq 0.59; d\alpha = 0.06;$$

$$E_{1,\alpha} = \{0.2; 0.3; 0.4; 0.5; 0.6; 0.7; 0.8; 0.9; 1\}; M(E_{1,\alpha}) = 0.6;$$

$$0.59 \leq \alpha \leq 0.6; d\alpha = 0.01;$$

$$E_{1,\alpha} = \{0.3; 0.4; 0.5; 0.6; 0.7; 0.8; 0.9; 1\}; M(E_{1,\alpha}) = 0.65;$$

$$0.6 \leq \alpha \leq 0.66; d\alpha = 0.06;$$

$$E_{1,\alpha} = \{0.3; 0.4; 0.5; 0.6; 0.7; 0.8; 0.9\}; M(E_{1,\alpha}) = 0.6;$$

$$0.6 \leq \alpha \leq 0.66; d\alpha = 0.06;$$

$$E_{1,\alpha} = \{0.4; 0.5; 0.6; 0.7; 0.8; 0.9\}; M(E_{1,\alpha}) = 0.65;$$

$$0.7 \leq \alpha \leq 0.75; d\alpha = 0.05;$$

$$E_{1,\alpha} = \{0.4; 0.5; 0.6; 0.7; 0.8\}; M(E_{1,\alpha}) = 0.6;$$

$$0.75 \leq \alpha \leq 0.8; d\alpha = 0.05;$$

$$E_{1,\alpha} = \{0.5; 0.6; 0.7; 0.8\}; M(E_{1,\alpha}) = 0.65;$$

$$0.8 \leq \alpha \leq 0.85; d\alpha = 0.05;$$

$$E_{1,\alpha} = \{0.5; 0.6; 0.7\}; M(E_{1,\alpha}) = 0.6;$$

$$0.85 \leq \alpha \leq 0.9; d\alpha = 0.05;$$

$$E_{1,\alpha} = \{0.6; 0.7\}; M(E_{1,\alpha}) = 0.65;$$

$$0.9 \leq \alpha \leq 0.96; d\alpha = 0.06;$$

$$E_{1,\alpha} = \{0.6\}; M(E_{1,\alpha}) = 0.6.$$

We find the expected values  $E_1$ :

$$\begin{aligned}
 F(E_1) &= \frac{1}{\alpha_{\max}} \int_0^{\alpha_{\max}} M(E_{1,\alpha}) d\alpha = \frac{1}{0.96} \int_0^{0.96} M(E_{1,\alpha}) d\alpha = \\
 &= 1/0.96(0.5 \cdot 0.5 + 0.55 \cdot 0.03 + 0.6 \cdot 0.06 + 0.65 \cdot 0.01 + \\
 &+ 0.6 \cdot 0.06 + 0.65 \cdot 0.04 + 0.6 \cdot 0.05 + 0.65 \cdot 0.05 + 0.6 \cdot 0.05 + 0.65 \cdot 0.05 + 0.6 \cdot 0.06) = 0.554.
 \end{aligned}$$

For the second alternative,

$$\begin{aligned}
 E_2 &= \{0.4/0; 0.5/0.1; 0.54/0.2; 0.59/0.3; 0.66/0.4; 0.7/0.5; 0.7/0.6; \\
 &0.7/0.7; 0.7/0.8; 0.7/0.9; 0.6/1\}.
 \end{aligned}$$

We compute the restrictive set:

$$0 < \alpha \leq 0.4; d\alpha = 0.4;$$

$$E_{2,\alpha} = \{0; 0.1; 0.2; 0.3; 0.4; 0.5; 0.6; 0.7; 0.8; 0.9; 1\}; M(E_{2,\alpha}) = 0.5;$$

$$0.4 \leq \alpha \leq 0.5; d\alpha = 0.1;$$

$$E_{2,\alpha} = \{0.1; 0.2; 0.3; 0.4; 0.5; 0.6; 0.7; 0.8; 0.9; 1\}; M(E_{2,\alpha}) = 0.55;$$

$$0.5 \leq \alpha \leq 0.54; d\alpha = 0.04;$$

$$E_{2,\alpha} = \{0.2; 0.3; 0.4; 0.5; 0.6; 0.7; 0.8; 0.9; 1\}; M(E_{2,\alpha}) = 0.6;$$

$$0.54 \leq \alpha \leq 0.59; d\alpha = 0.05;$$

$$E_{2,\alpha} = \{0.3; 0.4; 0.5; 0.6; 0.7; 0.8; 0.9; 1\}; M(E_{2,\alpha}) = 0.65;$$

$$0.59 \leq \alpha \leq 0.6; d\alpha = 0.01;$$

$$E_{1,\alpha} = \{0.4; 0.5; 0.6; 0.7; 0.8; 0.9; 1\}; M(E_{2,\alpha}) = 0.7;$$

$$0.6 \leq \alpha \leq 0.66; d\alpha = 0.06;$$

$$E_{2,\alpha} = \{0.4; 0.5; 0.6; 0.7; 0.8; 0.9\}; M(E_{2,\alpha}) = 0.65;$$

$$0.66 \leq \alpha \leq 0.7; d\alpha = 0.04;$$

$$E_{2,\alpha} = \{0.5; 0.6; 0.7; 0.8; 0.9; 1\}; M(E_{1,\alpha}) = 0.75.$$

And the expected value  $E_2$ :

$$F(E_2) = 1/0.7(0.5 \cdot 0.4 + 0.55 \cdot 0.1 + 0.6 \cdot 0.04 + 0.65 \cdot 0.05 + \\ + 0.7 \cdot 0.01 + 0.65 \cdot 0.06 + 0.75 \cdot 0.04) = 0.554.$$

For the third alternative:

$$E_3 = \{1/0; 1/0.1; 1/0.2; 1/0.3; 1/0.4; 1/0.5; 0.9/0.6; 0.8/0.7; 0.7/0.8; 0.6/0.9; 0.5/1\}$$

$$F(E_3) = 1/1(0.5 \cdot 0.5 + 0.45 \cdot 0.1 + 0.4 \cdot 0.1 + 0.35 \cdot 0.1 + 0.3 \cdot 0.1 + 0.25 \cdot 0.1) = 0.448.$$

for the fourth:

$$E_4 = \{0.9/0; 0.91/0.1; 0.9/0.2; 0.8/0.3; 0.7/0.4; 0.6/0.5; 0.5/0.6; \\ 0.4/0.7; 0.3/0.8; 0.2/0.9; 0.1/1\}$$

$$F(E_4) = 1/0.91(0.5 + 0.45 + 0.4 + 0.35 + 0.3 + 0.25 + 0.2 + 0.15 + 0.1 \cdot 0.1 + 0.1 \cdot 0.01) = \\ = 0.298.$$

And finally, for the fifth:

$$E_5 = \{0.7/0; 0.8/0.1; 0.9/0.2; 0.99/0.3; 0.9/0.4; 0.8/0.5; 0.7/0.6; 0.6/0.7; \\ 0.5/0.8; 0.4/0.9; 0.3/1\}$$

$$F(E_5) = 1/0.99(0.5 \cdot 0.3 + (0.45 + 0.4 + 0.35 \cdot 0.3 \cdot 3) \cdot 0.1 + 0.3 \cdot 0.09) = 0.391.$$

Thus the expected value for satisfactory alternative  $u_1$  is equal to 0.554 for  $u_2$  is equal to 0.554,  $u_3$  equals 0.425,  $u_4$  equals 0.298, and finally  $u_5$  equals 0.391.

As the best alternative we choose the one with the highest score.

### 3.9. Ranking alternatives based on heuristic approach

Consider an alternative ordering problem based on criteria that take into account fuzzy estimates and heuristic assumptions about the usefulness of linguistic assessment (see in Efstahiou, Rajkovitz 1980, Skalna et al. 2015, Terano et al. 1994, Turksen et al. 1992).

Decision-making process model includes: evaluation of alternatives, the submission of the decision maker, a description of the decision making process.

Each alternative can be described using the quality criteria  $D_i$ ,  $i = 1, 2, \dots, n$ . The domain of definition is represented as a Cartesian product of  $n$ -sets  $D_i$ ,  $D = D_1 \times D_2 \times \dots \times D_n$ . A point is defined as a respective set of  $n$  values  $(d_1, d_2, \dots, d_n) \in D$  where  $D_i \in D_i$  and is denoted by  $d^{(n)}$ .

If the criteria values are determined inaccurate (i.e. fuzzy), then alternative  $A$  is a fuzzy subspace of  $D$  and it is represented as a Cartesian product of fuzzy sets for  $D_i$

$$A = F(D_1) \times F(D_2) \times \dots \times F(D_n).$$

Where  $F(D_i)$  is an any subset of  $D_i$  with the membership function  $\mu_A(d^{(n)})$ . It is in accordance with the usual definition of fuzzy relation and fuzzy composition

$$\mu_A(d^{(n)}) = \min_{i=1, \dots, n} \mu_A(d_i).$$

Where  $\mu_A(d_i)$  is a function on the criterion of evaluation of alternative capabilities  $D_i$ .

The set of all possible alternatives is denoted by  $A$  and is a variety of fuzzy sets.

Universal elements of  $U$  are selected according to certain simple rules, for example,

$$U = \{\text{high, sufficiently high, medium, low enough, low}\}.$$

The dictionary can be expanded by introducing modifiers, such as “very quiet”, “more or less”.

Utility  $U$  can be interpreted as a linguistic variable, whose value is a term of the fuzzy sets, defined on the interval  $[0, 1]$ . Knowledge of the specified fuzzy utility ratio  $F$  of  $D = \{d_i^{(n)}\}$  at the union  $U = \{u\}$ , which is a fuzzy set in the Cartesian product  $D \times U$ , is very important.  $F$  is characterized by using the  $\mu_\Phi(d^{(n)}, u)$ , whereby each pair is assigned a value in the interval  $[0, 1]$ .  $F$  ratio is usually represented in a table giving different points of the utility  $D$ .

The knowledge of the utility as a fuzzy relation  $F$  makes it possible to characterize alternative  $A$  as a fuzzy subset  $V \in U$  where  $V = A \circ \Phi$ , using the relation

$$\mu_V(u) = \max\left(\min\left(\mu_\Phi(d^{(n)}, u), \mu_A(u)\right)\right).$$

According to the definition of the set  $V$ , each alternative  $A$  is more than one utility value, which has a different degrees of membership. For ranking the alternatives, it is

necessary to establish their order, using the best estimate equivalent to worse, as well as, find out how one fuzzy set is better than another.

Sometimes the highest degree of membership may be taken as a typical representative of the alternatives comparison. By arranging the set of alternatives in order of their usefulness,  $U$  elements are created. It expresses set  $V$  linguistically. Additionally,  $V$  can be expressed graphically, which allows compare alternatives visually.

**Example.** Let the dean's office be able to give the award to the students. Selecting a candidate for the award makes two interested parties involve: the dean's office and the Group's feature, expresses their preferences for candidates. Their statements define a set of performance criteria and targets, in particular:

- student achievement (performance)  $D_1 = \{\text{Excellent, Good, Satisfactory}\}$ ,
- student public activity  $D_2 = \{\text{Very high, High, Acceptable, Low}\}$ ,
- student discipline (attendance)  $D_3 = \{\text{Good, Acceptable, Poor}\}$ ,
- student income (incoming?)  $D_4 = \{\text{Good, Acceptable, Poor}\}$ .

The universal set for the utility is defined as follows

$$U = \{\text{high, sufficiently high, medium, low enough, low}\}.$$

You can also use the modifier "and", "or", "very". There is defined utility based on a survey of the dean's office representatives and the group's property value of the linguistic groups. Basic utility variable belongs to the interval  $[0, 1]$ .

Heuristics dean's office:

1. Excellent academic performance is more preferably than good academic performance.
2. Activity may be acceptable if there is a good performance. Although the activity must be very high, if the performance is good.
3. Discipline should be good.
4. Incoming students are not taken into account.

Heuristics property groups:

1. Progress should not be satisfactory.
2. Usefulness at an acceptable high activity, excellent performance and a poor material position; the usefulness of a sufficiently high at a very high or a high activity, a good performance, a good financial position.
3. Evaluation of discipline is not considered.
4. With a good material position of candidate there is taken into account only confirmed excellent performance and very high activity.

So we can obtain

**The ideal candidate in terms of the dean’s office:**

Excellent performance, very high activity, good discipline.

**The ideal candidate in terms of the Group’s feature:**

Excellent performance, very high activity, the poor financial situation.

**Construction of utility relations.** The solution space  $D = D_1 \times D_4 \times D_3 \times D_4$  comprises  $3 \times 4 \times 3 \times 3 = 108$   $n$ -dimensional tuples. Using heuristics, those elements, which are considered insignificant for both groups according to criteria performance evaluation, are rejected and we obtain a reduced set of criteria as follows

- $D_1 = \{\text{Excellent, Good}\}$
- $D_2 = \{\text{Very high, High, Satisfactory}\}$
- $D_3 = \{\text{Good}\}$
- $D_4 = \{\text{Good, Poor}\}$

Thus, the number of  $n$ -dimensional tuples, covering an area of utility, is reduced to  $2 \times 3 \times 1 \times 2 = 12$  points ( $n$ -tuple – see Tab. 3.2). After that the group agree to assess the utility of  $n$ -tuples.

**Table 3.2**  
Relationship utility  $F$  for evaluating alternatives

Student achievement	Student public activity	Student discipline	Student incoming	Utility to the dean’s office	Utility group
EXCELLENT	VERY HIGH	GOOD	POOR	VERY VERY HIGH	VERY VERY VERY HIGH
EXCELLENT	VERY HIGH	GOOD	GOOD	VERY VERY HIGH	VERY VERY VERY HIGH
GOOD	VERY HIGH	GOOD	POOR	HIGH	VERY VERY HIGH
GOOD	VERY HIGH	GOOD	GOOD	HIGH	HIGH
EXCELLENT	HIGH	GOOD	POOR	VERY HIGH	VERY VERY HIGH
EXCELLENT	HIGH	GOOD	GOOD	VERY HIGH	HIGH
GOOD	HIGH	GOOD	POOR	HIGH	VERY HIGH
GOOD	HIGH	GOOD	GOOD	HIGH	ENOUGH HIGH
EXCELLENT	ACCEPTABLE	GOOD	POOR	HIGH ENOUGH	ENOUGH HIGH
EXCELLENT	ACCEPTABLE	GOOD	GOOD	HIGH ENOUGH	AVERAGE
GOOD	ACCEPTABLE	GOOD	POOR	AVERAGE	ENOUGH HIGH
GOOD	ACCEPTABLE	GOOD	GOOD	AVERAGE	AVERAGE



Suppose there are three candidates for the award, which can be characterized as follows:

1. Student with good academic performance, very high activity, good discipline and a poor financial situation

$$A_1 = \{\text{GOOD, VERY HIGH, GOOD, POOR}\}.$$

2. For the second activity of the student is evaluated rather as high than satisfactory. This can be expressed by a fuzzy set

$$F(D_2) = \{0.6 / \text{HIGH}; 0.4 / \text{SUFFICIENTLY}\}.$$

The values of the other criteria are crisp. The alternative is described as follows:

$$A_2 = \{0.6 / (\text{EXCELLENT, HIGH, GOOD, POOR}); 0.4 / (\text{EXCELLENT, SUFFICIENTLY, GOOD, POOR})\}.$$

3. Third candidate estimation can be written as follows:

$$F(D_1) = \{0.6 / \text{EXCELLENT}; 0.3 / \text{GOOD}\};$$

$$F(D_2) = \{0.2 / \text{HIGH}; 0.8 / \text{SUFFICIENTLY}\};$$

$$F(D_3) = \{1 / \text{GOOD}\}.$$

$$F(D_4) = \{0.6 / \text{GOOD}; 0.4 / \text{POOR}\}.$$

A third alternative is defined as the direct product of the sets, and the minimum is taken by the membership function, i.e.,

$$A_3 = \{\min \{0.6, 0.2, 1, 0.6\} / (\text{EXCELLENT, HIGH, GOOD, GOOD});$$

$$\min \{0.6, 0.8, 1, 0.6\} / (\text{EXCELLENT, SUFFICIENTLY, GOOD, GOOD});$$

$$\min \{0.6, 0.2, 1, 0.4\} / (\text{EXCELLENT, HIGH, GOOD, POOR});$$

$$\min \{0.6, 0.8, 1, 0.4\} / (\text{EXCELLENT, SUFFICIENTLY, GOOD, POOR});$$

$$\min \{0.3, 0.2, 1, 0.6\} / (\text{GOOD, HIGH, GOOD, GOOD});$$

$$\min \{0.3, 0.8, 1, 0.6\} / (\text{GOOD, SUFFICIENTLY, GOOD, GOOD});$$

$$\min \{0.3, 0.2, 1, 0.4\} / (\text{GOOD, HIGH, GOOD, POOR});$$

$$\min \{0.3, 0.8, 1, 0.4\} / (\text{GOOD, SUFFICIENTLY, GOOD, POOR})\}$$

Thus, we have

- $A_3 = \{0.2 / (\text{EXCELLENT, HIGH, GOOD, GOOD});$
- $0.6 / (\text{EXCELLENT, SUFFICIENTLY, GOOD, GOOD});$
- $0.2 / (\text{EXCELLENT, HIGH, GOOD, POOR});$
- $0.4 / (\text{EXCELLENT, SUFFICIENTLY, GOOD, POOR});$
- $0.2 / (\text{GOOD, HIGH, GOOD, GOOD});$
- $0.3 / (\text{GOOD, SUFFICIENTLY, GOOD, GOOD});$
- $0.2 / (\text{GOOD, HIGH, GOOD, POOR});$
- $0.3 / (\text{GOOD, SUFFICIENTLY, GOOD, POOR})\}$

The calculating alternative of utility is based on the description and tables alternatives utility of sets is described by the values of ratings for each of the groups. Let it be marked as  $V(A_i)$  as shown in Table 3.3.

**Table 3.3**  
Utility value in the form of fuzzy sets

Alternative	Dean's office	Group's feature
$A_1$	{HIGH}	{VERY-VERY-HIGH}
$A_2$	{0.6 / VERY HIGH; 0.4 / HIGH ENOUGH}	{0.6 / VERY, VERY HIGH; 0.4 / HIGH ENOUGH}
$A_3$	{0.2 / VERY HIGH; 0.6 / SUFFICIENTLY HIGH; 0.2 / VERY HIGH; 0.4 / SUFFICIENTLY HIGH; 0.2 / HIGH; 0.3 / MEDIUM; 0.2 / HIGH; 0.3 / SECONDARY}	{0.2 / HIGH; 0.6 / MEDIUM; 0.2 / VERY, VERY HIGH; 0.4 / SUFFICIENTLY HIGH; 0.2 / VERY HIGH; 0.3 / MEDIUM; 0.2 / SUFFICIENTLY HIGH; 0.3 / HIGH ENOUGH}

**Ordering alternatives.** Utility alternative is presented as a fuzzy set  $V$ . The next step is to rank the fuzzy sets and to establish a better alternative. Each term utility can be represented as a fuzzy set on the base variable  $U^* = [0, 1]$ . Element of the fuzzy set  $V$  can be represented as a fuzzy subset  $U^*$  instead of  $U$  that is, if

$$\text{HIGH} = \{1/1, 0.7/0.9, 0.3/0.8\}, \text{ then } 0.5 / \text{HIGH} = \{0.5/1, 0.5/0.9, 0.3/0.8\}.$$

When there is applied "rule of minimum", it preserves the commutative, that is, the order of calculations is irrelevant. This means that if  $V$  is calculated on the  $U$ , and

then translated into  $U^*$ , so the same value will be obtained only if it is simply to calculate  $V$  directly at  $U^*$ . After the transfer from  $U$  to  $U^*$ , the elements of  $V$  can be combined using the “rule of maximum”. In Table 3.4 the results are expressed linguistically for each alternative.

**Table 3.4**  
The results in the linguistic form

Alternative	Dean’s office	Group
$A_1$	HIGH	VERY, VERY HIGH
$A_2$	VERY HIGH or HIGH ENOUGH	VERY, VERY HIGH or HIGH ENOUGH
$A_3$	HIGH ENOUGH	MEDIUM or HIGH ENOUGH, but NOT VERY VERY HIGH

Fuzzy sets  $V$  (alternative utility) is useful to compare visually with the decision-makers group as it is shown in Figures 3.25 and 3.26.

Based on the obtained features it can be concluded that the best alternative is

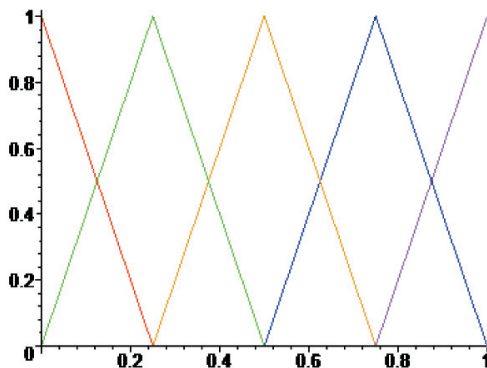
$A_1 = \{\text{GOOD performance, VERY HIGH activity, GOOD discipline POOR material position}\}$ .

This result was obtained from the ratio

$$\mu_{A \cap B}(u) = \min\{\mu_A(u), \mu_B(u)\},$$

where:

- $A$  – the dean’s office,
- $B$  – feature group,
- $u$  – useful to consider alternatives.



**Fig. 3.25.** Membership functions of linguistic utilities (low – red, low enough – green, medium – yellow, sufficiently high – blue, high – purple)

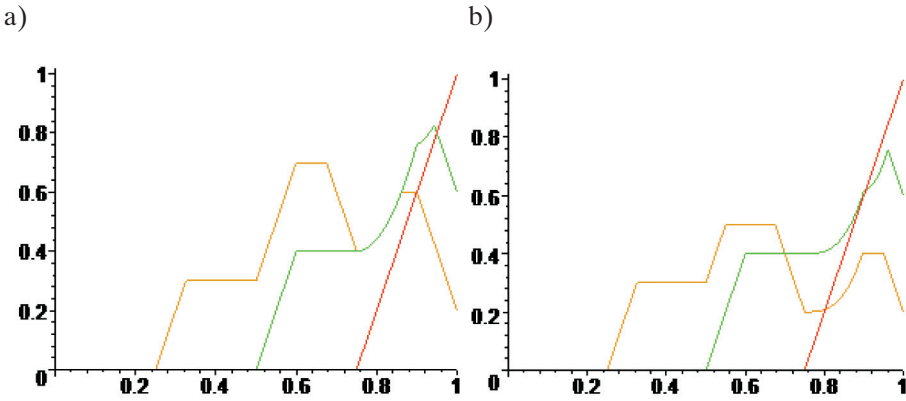


Fig. 3.26. Evaluating the usefulness of the dean’s office (a);  
evaluating the usefulness of the feature group (b)

### 3.10. Fuzzy decision trees

Decision tree is one of the most popular options for learning and reasoning based on examples. Specifically decision trees allow us to classify objects relying on attribute values, from which the decision rules can be extracted. The idea of building a decision tree is that the identified root node builds a path to the leaf nodes. Besides, in this case, the attributes associated with intermediate nodes are identified by entropy measures to determine the classification of the preference rating on this path. Every path to a leaf node generates a decision rule “if... then...” used to classify objects.

On the other hand, the maintenance of fuzzy sets and fuzzy logic (see in Zadeh 1965, 1996) allowed us to obtain a general methodology that admit to consider the concept of uncertainty and inaccuracy. Decision trees based on fuzzy set theory solutions combine the advantages of decision trees and the ability of a fuzzy representation to handle inaccurate and uncertain data see in Bezdek 1981, Borovik et al. 2013, *Building Classification Models: ID3...*, Chernov et al. 2016, 2015, Evans, Lohse 2011, Intan, Yuliana 2009, Olaru, Wehenkel 2003, Rockach, Maimon 2014, Smith 2002).

The hallmark of decision trees is the fact that every example of learning sample belongs to a particular node in the tree. However in the fuzzy decision tree, every example is non-deterministically assigned. For each attribute you must allocate some of its linguistic values and determine the membership degree in the examples of learning attempts. Instead, the number of individual nodes are examples of fuzzy decision

tree groups cases based on membership degree. The membership ratio of examples  $D_j \in S^N$  for nodes  $N$  and  $i$  to the target value, can be calculated as follows

$$P_i^N = \sum_{S^N} \min(\mu_N(D_j), \mu_i(D_j)) \quad (3.1)$$

where  $\mu_N(D_j)$  is a degree of membership of example  $D_j$  on the node  $N$ ,  $\mu_i(D_j)$  is an example degree accessories relative to the target value  $i$ ,  $S^N$  is the set of all node examples  $N$ . In this way, attributes are assigned to nodes based on the lowest level of uncertainty classification.

The next step will be to identify coefficient  $P^N$ , which is responsible for the general characteristics of the node  $N$ . In the example's standard decision tree algorithm determines the number ratio of instances belonging to a particular attribute, the total number of examples. For trees the indistinct ratio  $P_i^N/P^N$  is taken into account to calculate the degree of membership.

The expression shown below (3.2) gives the estimated average information number for determining a class object from a plurality of  $P^N$ .

$$E(S^N) = - \sum_i \frac{P_i^N}{P^N} \cdot \log_2 \frac{P_i^N}{P^N} \quad (3.2)$$

The next step (3.3) is the construction of a fuzzy decision tree to find the partition entropy for the attribute  $A$  with the values  $a_j$

$$E(S^N, A) = \sum_j \frac{P^{N|j}}{P^N} \cdot E(S^{N|j}) \quad (3.3)$$

where the node  $N|j$  is a child of the node  $N$ .

The process of constructing the tree begins with finding the condition of attribute that defines a root node (here and below, this process is repeated iteratively). For determining one node, you need to calculate a classification of each attribute of ambiguity conditions. Determining the classification uncertainty of fuzzy partition is reduced to selecting attribute  $A^*$  with a maximum gain of information as shown in equations:

$$G(S^N, A) = E(S^N) - E(S^N, A) \quad (3.4)$$

$$A^* = \arg \max_A G(S, A) \quad (3.5)$$

Node  $N$  is divided into several sub-assemblies  $N|j$ . The degree of membership  $D_k$  example to the node  $N|j$  is calculated incrementally from the node  $N$  as follows

$$\mu_{N|j}(e_k) = \min(\mu_{N|j}(D_k), \mu_{N|j}(D_k, a_j)) \quad (3.6)$$

where  $\mu_{N|j}(D_k, a_j)$  shows the degree of membership  $D_k$  to attribute  $a_j$ .

Subassembly  $N|j$  is deleted if all the examples have degree of membership equal zero. The algorithm is repeated for as long as all node examples are not classified or all the attributes are used.

Belonging to the target class for the new data record is expressed by the following equality

$$\sigma_j = \frac{\sum_l \sum_k P_k^l \cdot \mu_l(D_j) \cdot \chi_k}{\sum_l (\mu_l(D_j) \cdot \sum_k P_k^l)} \quad (3.7)$$

here:

- $P_k^l$  – the correlation coefficient examples tree leaf  $\lambda$  for the target class of  $k$  values,
- $\mu_l(D_j)$  – the degree of membership to the node example  $\lambda$ ,
- $\chi_k$  – belonging the target class value  $k$  to the positive (or negative) value classification outcome.

**Consider an example.** Let's consider the situation, when applying for a job, applicants are interviewed. The result of the interview is an evaluation of the skill level (Qualification) on a scale from 0 (no knowledge and experience for the post, claimed by the candidate) to 10 (Expert). The second point refers to the level of salary (Salary), which a candidate would get for the work. The third item is a rating of the applicant.

Let's describe the attributes : "Qualification" can be assigned with "Beginner", "Specialist" and "Expert" and the attribute "Salary" can be denoted as "Low", "Medium" and "High".

The fuzzy values of the candidate's skills levels are shown in the Figure 3.27. The fuzzy values of candidate's expectation of salary are presented in the Figure 3.28.

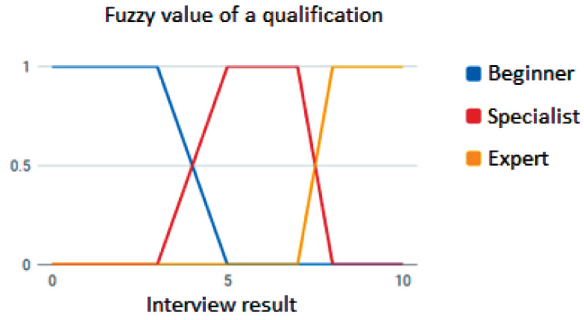


Fig. 3.27. The fuzzy values of qualifications

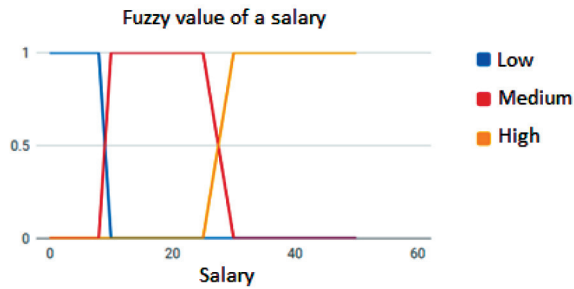


Fig. 3.28. The fuzzy values of salary expectations

Suppose we have a training set of  $n = 7$  candidates (7 rows of data see in Tab. 3.5).

**Table 3.5**  
Training sample

Number	Qualification	Salary	Rating
0	0	9	0
1	1	16	0
2	4	22	0.1
3	4.5	27	0.3
4	6	17	0.7
5	7.7	28	0.9
6	9	50	1

We must construct a fuzzy decision tree and find the challenger ranking with 4 qualification and salary level of 25 thousand dollars.

The first step, shown in Table 3.6, is to find a degree of membership referring to examples of training sample attributes, that is to find the appropriate value for a function  $\mu_i^{\text{rate}}(D_j)$  and  $\mu_i^{\text{pay}}(D_j)$ .

**Table 3.6**  
Values for a function  $\mu_i^{\text{rate}}(D_j)$  and  $\mu_i^{\text{pay}}(D_j)$

Number	Qualification			Salary		
	“Beginner”	“Specialist”	“Expert”	“Low”	“Average”	“High”
0	1	0	0	0.5	0.5	0
1	1	0	0	0	1	0
2	0.5	0.5	0	0	0.8	0.2
3	0.25	0.75	0	0	0.3	0.7
4	0	1	0	0	1	0
5	0	0.3	0.7	0	0.2	0.8
6	0	0	1	0	0	1
$\Sigma$	2.75	2.55	1.7	0.5	3.8	2.7

Here  $\mu_0^{\text{rate}}(D_j)$  corresponds to “Beginner” qualification attribute for the  $j$ -th data,  $\mu_1^{\text{rate}}(D_j)$  conform to “Specialist” and  $\mu_2^{\text{rate}}(D_j)$  to “Expert”. Consequently  $\mu_0^{\text{pay}}(D_j)$  meaning “Low” by salary attribute for the  $j$ -th data,  $\mu_1^{\text{pay}}(D_j)$  – “Average” and  $\mu_2^{\text{pay}}(D_j)$  – “High”.

We calculate the membership function of belonging participation to successful outcome (sum of rating from Tab. 3.5)

$$DAM = 0 + 0 + 0.1 + 0.3 + 0.7 + 0.9 + 1 = 3,$$

and unsuccessful

$$R_{NOT} = (1-0) + (1-0) + (1-0.1) + (1-0.3) + (1-0.7) + (1-0.9) + (1-1) = 4.$$

The value of the total entropy

$$E(S^N) = -\frac{P_{\text{yes}}}{P_{\text{yes}} + P_{\text{no}}} \log_2 \frac{P_{\text{yes}}}{P_{\text{yes}} + P_{\text{no}}} - \frac{P_{\text{no}}}{P_{\text{yes}} + P_{\text{no}}} \log_2 \frac{P_{\text{no}}}{P_{\text{yes}} + P_{\text{no}}},$$



is equal to

$$E(S^N) = -\frac{3}{7} \log_2 \frac{3}{7} - \frac{4}{7} \log_2 \frac{4}{7} \approx 0.985.$$

Now let's calculate  $E(S^N, \text{"Qualifications"})$ :

$$\begin{aligned} P_{\text{yes}}^{\text{beginner}} &= \sum_j \min(d_{2,j}, \mu_0^{\text{rate}}(D_j)) = \min(0, 1) + \min(0, 1) + \min(0.1; 0.5) + \\ &+ \min(0.3; 0.25) + \min(0.7, 0) + \min(0.9, 0) + \min(1, 0) = 0 + 0 + 0.1 + \\ &+ 0.25 + 0 + 0 + 0 = 0.35, \end{aligned}$$

$$\begin{aligned} P_{\text{no}}^{\text{beginner}} &= \sum_j \min(1 - d_{2,j}, \mu_0^{\text{rate}}(D_j)) = \min(1, 1) + \min(1, 1) + \min(0.9; 0.5) + \\ &\min(0.7; 0.25) + \min(0.3, 0) + \min(0, 1, 0) + \min(0, 0) = 1 + 1 + 0.5 + \\ &+ 0.25 + 0 + 0 + 0 = 2.75, \end{aligned}$$

$$P^{\text{beginner}} = 0.35 + 2.75 = 3.1.$$

Next we calculate the entropy corresponding to the skill level "beginner":

$$\begin{aligned} E(\text{Qualification, Beginner}) &= -\frac{P_{\text{yes}}^{\text{beginner}}}{P_{\text{yes}}^{\text{beginner}} + P_{\text{no}}^{\text{beginner}}} \log_2 \frac{P_{\text{yes}}^{\text{beginner}}}{P_{\text{yes}}^{\text{beginner}} + P_{\text{no}}^{\text{beginner}}} - \\ &-\frac{P_{\text{no}}^{\text{beginner}}}{P_{\text{yes}}^{\text{beginner}} + P_{\text{no}}^{\text{beginner}}} \log_2 \frac{P_{\text{no}}^{\text{beginner}}}{P_{\text{no}}^{\text{beginner}} + P_{\text{no}}^{\text{beginner}}} = \\ &= -\frac{0.35}{3.1} \log_2 \frac{0.35}{3.1} - \frac{2.75}{3.1} \log_2 \frac{2.75}{3.1} \approx 0.5086. \end{aligned}$$

Further, for the skill level "specialist":

$$P_{\text{yes}}^{\text{specialist}} = \sum_j \min(d_{2,j}, \mu_1^{\text{rate}}(D_j)) = 1.4,$$

$$P_{\text{no}}^{\text{specialist}} = \sum_j \min(1 - d_{2,j}, \mu_1^{\text{rate}}(D_j)) = 1.6,$$

$$P^{\text{specialist}} = 1.4 + 1.6 = 3,$$

$$E(\text{Qualification, Specialist}) = -\frac{1.4}{3} \log_2 \frac{1.4}{3} - \frac{1.6}{3} \log_2 \frac{1.6}{3} \approx 0.997.$$

At the end for the skill level “expert”:

$$P_{\text{yes}}^{\text{expert}} = \sum_j \min(d_{2,j}, \mu_2^{\text{rate}}(D_j)) = 1.7,$$

$$P_{\text{no}}^{\text{expert}} = \sum_j \min(1 - d_{2,j}, \mu_2^{\text{rate}}(D_j)) = 0.1,$$

$$P^{\text{expert}} = 1.8,$$

$$E(\text{Qualification, Expert}) = -\frac{1.7}{1.8} \log_2 \frac{1.7}{1.8} - \frac{0.1}{1.8} \log_2 \frac{0.1}{1.8} \approx 0.3095.$$

Thus, the entropy values for “Qualification” attribute are shown in Table 3.7.

**Table 3.7**

The juxtaposition relating to entropy of “Qualification” attribute

	“Beginner”	“Specialist”	“Expert”
$R_{\text{yes}}$	0.35	1.4	1.7
$R_{\text{no}}$	2.75	1.6	0.1
Entropy	0.509	0.997	0.31

The next step is the calculating the entropy for the attribute “Qualification”

$$\begin{aligned} E(S^N, \text{Qualification}) &= \frac{\sum_0^{\text{rate}}}{P_{\text{yes}} + P_{\text{no}}} E^{\text{beginner}} + \frac{\sum_1^{\text{rate}}}{P_{\text{yes}} + P_{\text{no}}} E^{\text{specialist}} + \frac{\sum_2^{\text{rate}}}{P_{\text{yes}} + P_{\text{no}}} E^{\text{expert}} = \\ &= \frac{2.75}{7} 0.509 + \frac{2.55}{7} 0.997 + \frac{1.7}{7} 0.31 \approx 0.638. \end{aligned}$$

On the last phase we calculate the information gain for a given attribute.

$$G(S^N, \text{Qualification}) = E(S^N) - E(S^N, \text{Qualification}) \approx 0.985 - 0.638 = 0.347.$$

Likewise we find the entropy values for “salary” attribute (see Tab. 3.8).

**Table 3.8**

The juxtaposition relating to entropy of “Salary” attribute

	“Low”	“Middle”	“High”
$R_{\text{yes}}$	0	1.3	2.2
$R_{\text{no}}$	0.5	3	1
Entropy	0	0.884	0.896

The entropy value for the attribute “Salary” is calculating as follows

$$\begin{aligned}
 E(S^N, \text{Salary}) &= \frac{\sum_0^{\text{pay}}}{P_{\text{yes}} + P_{\text{no}}} E^{\text{low}} + \frac{\sum_1^{\text{pay}}}{P_{\text{yes}} + P_{\text{no}}} E^{\text{middle}} + \frac{\sum_2^{\text{pay}}}{P_{\text{yes}} + P_{\text{no}}} E^{\text{high}} = \\
 &= \frac{0.5}{7} \cdot 0 + \frac{3.8}{7} \cdot 0.884 + \frac{2.7}{7} \cdot 0.896 \approx 0.82556.
 \end{aligned}$$

Now, we calculate the information gain for a given attribute in the following form

$$G(S^N, \text{Salary}) = E(S^N) - E(S^N | \text{Salary}) \approx 0.985 - 0.826 = 0.159.$$

The maximum of gathered information appoints to the attribute “Qualification”, therefore, the partition will start with this attribute and the root node is “Qualified”.

The next step is for each value of the training sample to obtain a membership degree to each node of the tree. For this purpose, the equation (3.6) is used:  $\mu_{N|j}(e_k) = \min(\mu_{N|j}(D_k), \mu_{N|j}(D_k, a_j))$ :

$$\mu_{\text{beginner}|\text{low}}(e_0) = \min(\mu_0^{\text{rate}}(D_0), \mu_0^{\text{pay}}(D_0)) = \min(1, 0.5) = 0.5,$$

$$\mu_{\text{beginner}|\text{middle}}(e_0) = \min(\mu_0^{\text{rate}}(D_0), \mu_1^{\text{pay}}(D_0)) = \min(1, 0.5) = 0.5,$$

$$\mu_{\text{beginner}|\text{high}}(e_0) = \min(\mu_0^{\text{rate}}(D_0), \mu_2^{\text{pay}}(D_0)) = \min(1, 0) = 0,$$

$$\mu_{\text{specialist}|\text{low}}(e_0) = \min(\mu_1^{\text{rate}}(D_0), \mu_0^{\text{pay}}(D_0)) = \min(0, 0.5) = 0,$$

...

$$\mu_{\text{expert}|\text{high}}(e_n) = \min(\mu_2^{\text{rate}}(D_n), \mu_2^{\text{pay}}(D_n)) = \min(1, 1) = 1.$$

Thus, we receive the following set of data training samples (see Tab. 3.9) belonging to each tree node.

**Table 3.9**  
The data training after division

“Qualification”	“Beginner”			“Specialist”			“Expert”		
“Salary”	low	middle	high	low	middle	high	low	middle	high
0	0.5	0.5	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0
2	0	0.5	0.2	0	0.5	0.2	0	0	0
3	0	0.25	0.25	0	0.3	0.7	0	0	0
4	0	0	0	0	1	0	0	0	0
5	0	0	0	0	0.2	0.3	0	0.2	0.7
6	0	0	0	0	0	0	0	0	1

For each node we can determine coefficients  $P_i^N$  in the following form

$$P_i^N = \sum_{S^N} \min(\mu_N(D_j), \mu_i(D_j)).$$

For nodes with a parent root, the node has

$$P_{\text{yes}}^{\text{beginner}} = \min(\mu_{\text{beginner}|\text{low}}(e_0), d_{2,0}) + \min(\mu_{\text{beginner}|\text{middle}}(e_0), d_{2,0}) + \\ + \min(\mu_{\text{beginner}|\text{high}}(e_n), d_{2,n}) = \min(0.5, 0) + \min(0.5, 0) + \dots + \min(0, 1) = 0.7,$$

$$P_{\text{no}}^{\text{beginner}} = \min(\mu_{\text{beginner}|\text{low}}(e_0), 1 - d_{2,0}) + \min(\mu_{\text{beginner}|\text{middle}}(e_0), 1 - d_{2,0}) + \\ + \min(\mu_{\text{beginner}|\text{high}}(e_n), 1 - d_{2,n}) = \min(0.5, 1) + \min(0.5, 1) + \dots + \min(0, 0) = 3.2,$$

etc.

$$P_{\text{yes}}^{\text{specialist}} = 2, \quad P_{\text{no}}^{\text{specialist}} = 2.2, \quad P_{\text{yes}}^{\text{expert}} = 1.9, \quad P_{\text{no}}^{\text{expert}} = 0.2.$$

For nodes whose parent is “Beginner”, we get now:

$$P_{\text{yes}}^{\text{beginner}|\text{low}} = \sum_{i=0}^{n-1} \min(\mu_{\text{beginner}|\text{low}}(e_i), d_{2,i}) = 0,$$

$$P_{\text{no}}^{\text{beginner}|\text{low}} = \sum_{i=0}^{n-1} \min(\mu_{\text{beginner}|\text{low}}(e_i), 1 - d_{2,i}) = 0.$$

and so on.

The result is a set shown in Table 3.10.

**Table 3.10**  
The juxtaposition relating to entropy of attributes

	“Beginning”	“Specialist”	“Expert”
“Low”	0 / 0.5	0 / 0	0 / 0
“Average”	0.35 / 2.25	1.3 / 1.2	0.2 / 0.1
“High”	0.35 / 0.45	0.7 / 1	1.7 / 0.1

The final result can be demonstrated on the following figures (Fig. 3.28) as a decision tree.

It should be reminded that we need to find a person with level 4 qualifications and a salary of 25 thousand dollars. First of all, we find the values of the membership functions referring to the applicant:

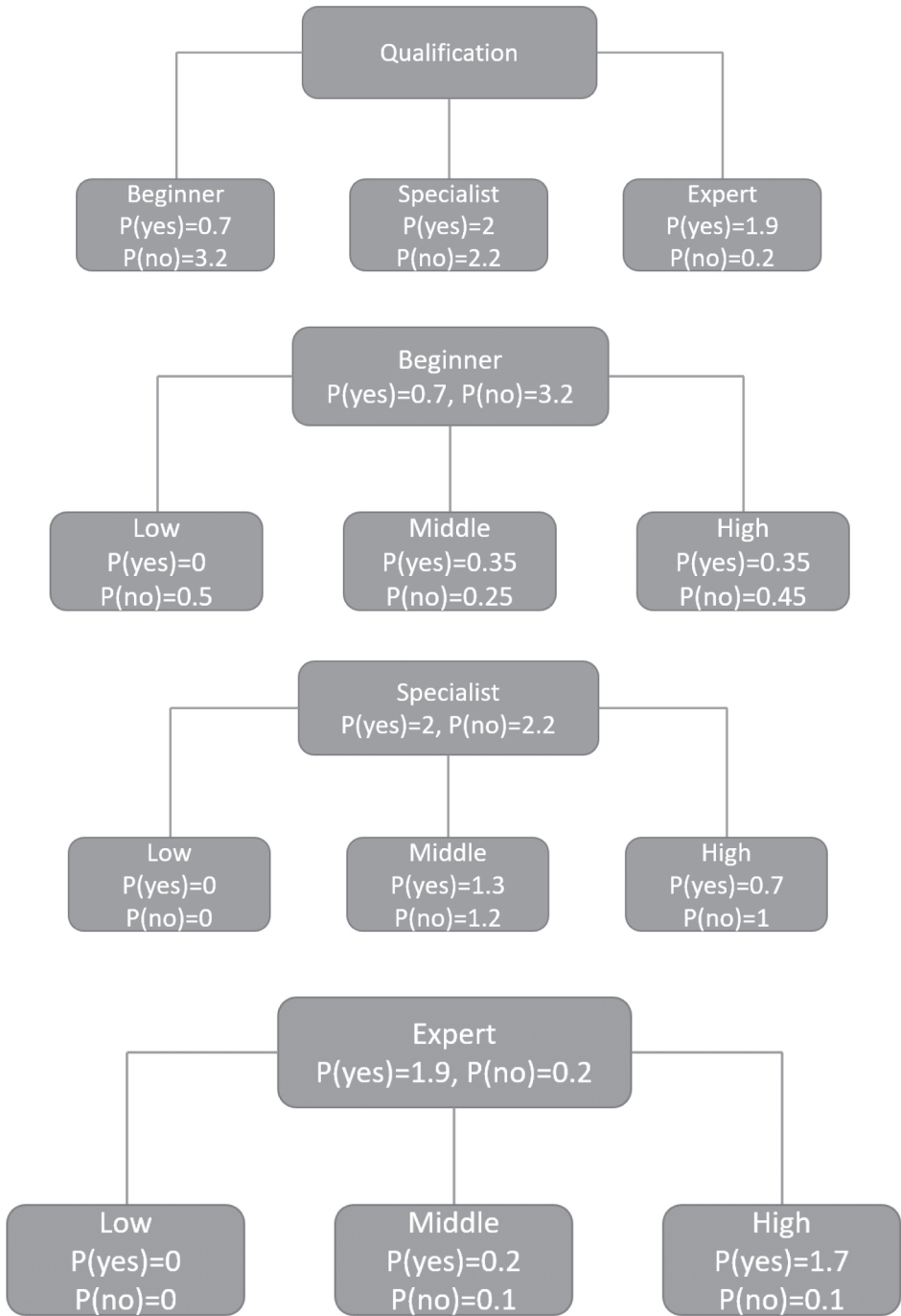
$$\mu_0^{\text{rate}}(\tilde{D}) = 0.5; \mu_1^{\text{rate}}(\tilde{D}) = 0.5; \mu_2^{\text{rate}}(\tilde{D}) = 0,$$

and

$$\mu_0^{\text{pay}}(\tilde{D}) = 0; \mu_1^{\text{pay}}(\tilde{D}) = 0.5; \mu_2^{\text{pay}}(\tilde{D}) = 0.5,$$

which corresponds to the membership of nodes:

- “Qualifications” = “Beginner” | “Salary” = “Middle”,
- “Qualifications” = “Beginner” | “Salary” = “High”,
- “Qualifications” = “Expert” | “Salary” = “Middle”,
- “Qualifications” = “Expert” | “Salary” = “High”.



**Fig. 3.28.** The final result as the decision tree

To find the rating use equation (3.7).

$$\tilde{\sigma} = \frac{\sum_l \sum_k P_k^l \cdot \mu_l(\tilde{D}) \cdot \chi_k}{\sum_l (\mu_l(\tilde{D}) \cdot \sum_k P_k^l)},$$

here  $\chi_k$  is an affiliation of the target class  $k$  value to a positive (or negative) value of the classification outcome, in this case, it may be a decision on hiring  $\chi_{\text{yes}} = 1$  and, consequently, the refusal  $\chi_{\text{no}} = 0$ .

In our case, we get

$$\begin{aligned} \sum_l \sum_k P_k^l \cdot \mu_l(\tilde{D}) \cdot \chi_k &= \sum_{i=0}^2 \mu_i^{\text{rate}} \cdot \sum_{j=0}^3 \mu_j^{\text{pay}} (P_{\text{yes}}^{ilj} \chi_{\text{yes}} + P_{\text{no}}^{ilj} \chi_{\text{no}}) = \\ &= 0.5 \cdot 0 \cdot (0 \cdot 1 + 0.5 \cdot 0) + 0.5 \cdot 0.5 \cdot (0.35 \cdot 1 + 2.25 \cdot 0) + \\ &+ 0.5 \cdot 0.5 \cdot (0.35 \cdot 1 + 0.45 \cdot 0) + 0.5 \cdot 0 \cdot (0 \cdot 1 + 0 \cdot 0) + \\ &+ 0.5 \cdot 0.5 \cdot (1.3 \cdot 1 + 1.2 \cdot 0) + 0.5 \cdot 0.5 \cdot (0.7 \cdot 1 + 1 \cdot 0) + 0 \cdot 0 \cdot (0 \cdot 1 + 0 \cdot 0) \\ &+ 0 \cdot 0.5 \cdot (0.2 \cdot 1 + 0.1 \cdot 0) + 0 \cdot 0.5 \cdot (1.7 \cdot 1 + 0.1 \cdot 0) = 0.675 \end{aligned}$$

and correspondingly,

$$\sum_l \sum_k P_k^l \cdot \mu_l(\tilde{D}) = 1.9.$$

As a result, we obtain the value of 0.355 ranking. Thus, the applicant has excessive demands, and the employer should refuse to accept him as a potential worker.

## 4. Soft computing in data handling

### 4.1. Introduction to soft computing

The term “soft computing” was entered in 1994 by the founder of the field relating to fuzzy logic – Lotfi A. Zadeh (see Yager and Zadeh 1992, Zadeh 1975, Zadeh 1996). This specified term stands for the set of empirical, indistinct, approximate methods of solving tasks which do not have exact decision algorithms for polynomial time (which operating time is polynomial depending on the size of the input data). Now, it is accepted to enlist the following methods to soft computing:

- neural networks,
- evolutionary strategy,
- fuzzy logic,
- multiagent systems (intelligence of pack),
- different heuristic algorithms of finding solutions,
- chaos theory, etc.

Many of the algorithms used in this study refers to the intellectual category. But the term “intelligence” (and “artificial intelligence”) is so fuzzed away now that it would be desirable to take AI into account.

We assume that the term “intelligence” as a property is a way to solve intellectual problems. We assume that an intellectual task is a task for which solution like an accurate algorithm has been not invented yet. In other words, the intelligence is a property of the person, the computer or something else, allowing to create new algorithms.

Such a determination at once allows us to cut down on a set marketing (in negative sense of this word) i.e. the determinants of intelligence tied to laundry detergents,



drugs and another goods, etc. Also it is interesting to apply this determination to human professions as follows:

- Programmer. A programmer’s product is the program – an algorithm in pure form. I do not think that there will be someone who will be able to claim that this profession is not intellectual even though we need to adjust given program to input and output data. For example, in the program there is a piece of a code solving already an existing problem. The programmer takes it, changes several symbols (constants) and receives the duplicated code adapted to new subject domain. It is about a typical sin of copying/pasting. Let’s not argue on shortcomings of such approach here – we will look at it on the other hand. The copied algorithm has been improved. The meaning of such duplication, intellectuality is close to zero.
- Hacker. There are typical schemes when crimes are committed on knurled technology. In this case an impressive intelligence is not necessary. Such hackers usually create systems to break in sooner or later. But there are also “stars” among them who come up with the unusual and unrepeatable solutions. Actually, new algorithms have been continuously written. The hacker intelligence is much-needed all the time.
- Detective. Perhaps, one may say, that the profession of the detective and hackers makes synergy couple. Hackers’ job is more intellectual, while detective job calls for the more intelligence in order to deal with the criminals, to ask them questions and invent responses based on non-standard methods of search.
- Cashier (bank, shop, etc.). In this profession the invention of a new algorithm is somehow wrong. Everything needs to be done carefully, accurately and equally. I do not want to tell at all that cashiers are stupid people. To learn this work, the intelligence is necessary. Often considerable. But during the work “block” of algorithmization needs to be switched-off. Therefore I cannot call this profession intellectual according to the accepted determination.
- Driver. A combination of advance algorithmic activity (observance of traffic regulations, standard acceptances of driving) and permanent search for algorithms of the solving arising problems. Recognizing a road situation at a large number of the participating objects moving diversely with different speeds, searching for an optimum route at the movement in the unfamiliar place, responding to the changed traffic conditions in already familiar places (holes, repair of the road, new road signs). It is no wonder that this activity has not automated up to the end yet, there are only separate components helping to solve of separate tasks, for example, GPS navigators.

Thinking of intellectual features among professions (in the sense of using intelligence) can be finished – the reader can analyze the directions interesting to him on his

own. Also, we had an additional criterion for evaluation of intellectual activity that is: the more automated this field is, the more intellectual it becomes (it is temporary criterion while the human is the most intelligent “device” on Earth). By the way, there are attempts to automate difficult activity – it is possible not only to increase intelligence of the car, but also to reduce the intelligence of task. For example, it is rather difficult to automate cultivation and harvesting on any field on the open area completely – there are too many unknowns. However, if you prepare the field “in line” in advance, placing all plants in the greenhouse, then complexity of a task will be reduced. It will become more solvable. As a result, your intelligence will be demonstrated through a proposed task simplification.

Let’s go back to the methods called “soft computing”. The fact that they are associated with intellectual tasks is connected, most likely, with the fact that they have high universality and can be applied before developing the good specialized method of the task solution.

Other important general property of soft computing is property of adaptivity – fine tuning under a task. It increases the level of assessment(which is often also connected with intellectual activity) of the solution of a task and reduces complexity of a task for the researcher. Presently, it can work with a task like a black box (a system with an unknown algorithm whose essence can only be guessed based on external features) or a gray box (a system with the possibility of limited configuration by the user), without going into the subtleties of work related to the management system, in hope that these subtleties will be compensated by “intelligence” of the method.

For this reason, the ownership of the listed methods is a big plus for any researcher, the programmer, the scientist, the engineer, the analyst.

The consequence relating to the universality of the methods, is an opportunity to mix them. It is often possible to come across such systems where the neural network stands for a genetic algorithm. Indistinct qualifiers are a basis of individuals in multi-agent systems.

## **4.2. Evolutionary calculations**

### **4.2.1. General Introduction**

Evolutionary methods have been continuously driven by nature have been used for more than one billion years. Certainly, it is about natural evolution. This process, according to most of scientists, eventually gave us Homo Sapience used as an intellectuality standard. Certainly, there are people who do not agree with this point of view. Well, we will not argue with them. Even, if they are right, algorithms which are received

as a result of modeling of these “wrong” processes show the efficiency in many tasks. And that is enough to give the right to use these algorithms.

The first scientist who offered the harmonious theory of an origin of new types was Charles Darwin. In the textbook “Origin of species”, which came out in 1859, a basis of evolution fall into the following processes:

- variability – new individuals of population practically always differ from the parents a little (and it is sometimes strong);
- selection – natural or artificial selection eliminates unsuccessful options of changes; only the successful (more adapted) ones remain to live;
- heredity – the changes which happened in one of generations are inherited by the descendants.

The set of these driving forces also allows types to adapt themselves to the changing environment, to be improved and survive. But at the Darwin’s times, only the selection mechanism was clear. Then new signs appeared and as they were transferred to descendants, they became clear much later. In 1944, O. Avery, K. MacLeod and M. McCarthy (see Avery et al. 1944) published results of the researches. They proved that “the type of Deoxyribonucleic acid” is responsible for hereditary processes in organisms. What acid is, the world learned still later – on April 27, 1953 the well-known article of Watson and Shout was published in the Nature magazine, and the world heard about a two-chained spiral of DNA for the first time.

More than half a century have passed since the discovery, but still it is impossible to tell that we know everything about DNA functioning mechanism. We constantly learn about new and new details. That different sites of DNA are mutated with a different frequency. The fact that many genes can be present at DNA in different number of copies (copy number variation), and in products of one or another protein, sensitivity to various diseases may depend on it (see in Bentley and Wakefield 1996).

We are still getting familiar with natural processes, which from the very beginning attracted researchers who wanted to repeat similar process using the computer facilities. Presently, the huge number of different algorithms have been collected. Of course it is believed that transfer of the major directions, beaun with genetic algorithms and Holland’s classification systems (Holland 1994). For the first time they were published in the early sixties, but they gained the largest distribution after publicizing the book becoming classics – “Adaptation in natural and artificial systems” (Holland 1994). Next L.J. Fogel in 1966 (Fogel et al. 1966, Fogel 1999) applied one of the first evolutionary algorithm in practice. A tree-based genetic programming was introduced by N.L. Cramer in 1995 (Cramer 1995). These ideas gained further development in the works (see Banzhaf et al. 1998, Bentley and Wakefield 1996, Bisebroek 1999, De Jong and Spears 1991, 1992, Goldberg 1989, Goldberg and Sastry 2001, Koza 1992,

Mitchel 1996, Whitley 1994, 2001) devoted to evolutionary modeling. It was necessary to find a solution of a global extremum problem on the basis of a set of independent automatic machines. In result simultaneously processes of the birth, development and death of individuals were modeled. Also a big contribution to development of evolutionary calculations was made by Fogel et al. 1966.

Each of these schools took something special from the principles of evolution, known at that time. After that it was simplified to such an extent that process could be remodeled on the computer.

Along with many empirical algorithms, evolutionary algorithms do not guarantee finding a good result. Also they seem to be difficult when it comes to the use. Through selecting the wrong degeneration parameters, an incorrect search capability can be displayed. In this case you should not throw evolutionary algorithms at once – it is worth trying to help their natural intelligence (for example “play” with the settings). Here it is good to remember that if any type remains in very little amount– less than ten individuals, then in the nature it is doomed to extinction because of degeneration. But it is better to remember bodies of the birds, strange in shape and their wings, echolocators of dolphins and bats, the black mold using the radiation energy of the destroyed reactor.

#### **4.2.2. Genetic algorithm**

The genetic algorithm (further GA) is one of the most known evolutionary algorithms. It is simple in the principles of work, but has a high potential of development, as we will try to show in this section. More details and explanations can be found for instance in Fogel et al. 1966, Fogel 1999, De Jong 1975, De Jong and Spears 1991, 1992, Deb and Agrawal 1999, Duda and Szydło 2011, Goldberg 1989, Goldberg and Sastry 2001, Koza 1994, Mitchell 1996, Whitley 1994, 2001.

In essence, GA is an algorithm that is used for searching a global optimum of multiextremal function. For this purpose it uses, with a certain extent of approach, model of reproduction of live organisms. To solve a problem, we need to present it in the form of so-called fitness function from many variables (also called estimated)

$$f(x_1, x_2, x_3, \dots, x_N).$$

To solve the task, we need to find the global maximum or minimum (it is not fundamentally important which task we solve, because the search for the maximum can easily be replaced by the search for the minimum – and vice versa – for the same function – just include the opposite sign). At the same time certain restrictions are usually imposed on values of entrance variables, at least on the range of their change.

Before we consider the GA, we need to present all entrance variables in the form of chromosomes. Chromosomes in GA are meant as chains of symbols with which

further transactions are made. Most often apply the following two methods of parameters coding are applied (see in Janikow and Michalewicz):

- binary format,
- format from a floating comma.

When using a binary format, under parameter  $N$  bits are allocated (for each parameter  $N$  can be different). As for each of these parameters there are restrictions by MIN and MAX, mutual transition between parameter values in a format with a floating comma and their binary representation can be written down in the following type:

$$g = (r - \text{MIN}) / (\text{MAX} - \text{MIN}) \cdot (2^N - 1),$$

$$r = g \cdot (\text{MAX} - \text{MIN}) / (2^N - 1) + \text{MIN},$$

where:

- $g$  - the binary representation of parameter placed in  $N$  bits;
- $r$  - a parameter value in a format from a floating comma. We often use the subsequent transformation of the gained binary impression to Gray's code - it allows to reduce the destructive force of mutations (here we run a little forward).

The received binary representations of each parameter spread into a chain (line) of bits which farther is called a chromosome.

While working with parameters in a floating comma format, their value also gives the best in the bit chain, but without the transformation mentioned above, only directly in the representation with which the computer processor works.

After coding all necessary parameters in the form of a chromosome, we can start a basis cycle of a genetic algorithm:

1. Generation of initial accidental population.
2. Generation of the next generation.
3. Rejection of the worst decisions in again generated generation.
4. If do not reach criterion of the termination, we drop into a step 2.
5. Work completion. The copy which has the best value of fitness function is the required decision.

Here, of course, the stage 2 is the key. It can be detailed as follows:

1. Sort parent generation according to value of fitness function for each copy.
2. Enough copies of new generations have not generated yet:
  - 2.1. Select two parents.
  - 2.2. Combine their chromosomes (crossover).
  - 2.3. Apply other genetic operators.

Let's describe each of generation steps in details.

2.1. For selection different strategies are used. One of them is to choose  $N$  best copies in a random way. Another one is a tournament in a broad meaning. It is that for each of parents accidental couple (or more) of applicants is chosen. We choose from them the ones with the best value of fitness function. Thus, more adapted individuals will be more often parents. It is worth noticing that less adapted ones also have chance to pass.

To accelerate the convergence the strategy of elitism is also often used – in the next generation the best available solutions of the previous generation (elite) are adopted without changes. But with this approach it is necessary to be extremely careful – in case of insufficient size of population, it becomes similar to elite very quickly and search of new decisions practically stops.

2.2 and 2.3 – application of genetic operators. A basis of GA are two operators: the crossover which combines decisions of parents, and a mutation which provides search capabilities. Apart from these two main operators additional operators, for example, inversion can also be applied.

The operator of the crossover works with bit lines of two parent chromosomes. The simplest option is the single-point crossover. In this case each of parent chromosomes is cut in one, accidentally chosen point. The chromosome of the descendant is formed from “head” of a chromosome of one ancestor and “tail” of the other:

Ancestor 1:	1001101110101 100110	→	<u>100110111010</u>		<u>1010101</u>
Ancestor 2:	<u>0010110010110</u>  010101		1		2

The operation of the crossover can be more difficult – considering two-point crossover or even can use absolutely other principles (we still will return to it). The main thing is that combined ancestors' and descendants' decisions do not mean finding successful solutions again.

The operator of the mutation is simply a random chromosome change in one or more bits:

1001101110101100110      →      10011011**0**0101100110

Mutation – the destructive operator. In most cases, it violates the decision or even leads an individual to a disabled state. Therefore the probability of its application should not be excessively high. But lack of mutations nullifies capability of GA to search of a global optimum in all decision spaces.

Given as an example of additional operators, the operator of inversion consists of cyclic shift of bits in a chromosome that occurs accidental number of times

1001101110101100110 → 0110100110111010110

Now we have all components of a genetic algorithm, and we can fix them in practice.

### 4.2.3. Simple example of implementation of GA

Let's try to find a global extremum of one of test functions. It is known under the name DeJong 2 (see in DeJong 1975, DeJong and Spears 1991, DeJong and Spears 1992)

$$f(x, y) = \frac{100}{100 \cdot (x^2 - y)^2 + (x - 1)^2 + 1}.$$

This function is gully with rather small inclination around a maximum. The maximum of function is equal 100 at  $x = y = 1$  value. Of course, we will pretend that we do not know it, it is only known that the maximum is at parameter values (and “x” and “y”) somewhere between  $-1.28$  and  $+1.28$ .

The proposed program is concentrated on the algorithmization. Because of it, and in connection with an educational orientation of examples, the main settings are set in the text of programs in the form of constants. On a slang of programmers, parameters of this program are “hardcoded”. However it will be minus if the user of the program does not need to have access to a program code. We hope that such access will happen rather actively therefore editing of parameters in the text of the program can quite be considered a peculiar interface for the programmer. Thus, as the user's interface to change of parameters, we will use IDE for example, JetBrains PyCharm Community Edition with Python 3.7.

Let's define how we will represent a genome. In the first example we will apply a binary method relating to coding of a genome, without use of a Gray's code. We will store a chain of bits in the “int” type, and, in big-endian 16 bits will be responsible for parameter x, and little-endian – for parameter y. As value of fitness function on which it is necessary to make selection of the most adapted copies (sorting), conveniently a genome and this value is also tied to each genome to combine in one record.

Let's define the fields and constants used further:

```
import random

generation_size = 10000
generation_numbers = 200
mutation_probability = 0.2
```

The basis cycle of work of the program is concentrated in the Main method:

```
def __main__():
    generation = generateRandom()
    generation = sortGeneration(generation)
    for gen_num in range(1, generation_numbers):
        generation = generateNewGeneration(generation, True)
        generation = sortGeneration(generation)
        print("Best individual = ", weight(generation[0][0]))
        print("Generation = ", gen_num)
    print("x = ", getx(generation[0][0]))
    print("y = ", gety(generation[0][0]))
    print("Genome = ", hex(generation[0][0]))
```

Above we see a challenge of accidental initialization relating to starting generation and generation of the sequence of affiliated generations. The criterion for stopping creation of generations is reaching a certain number of generations. At the end of a cycle we remove parameters of the best copy in a console conclusion. As the collection of copies is sorted by fitness decrease, the copy with number 0 will be the best.

Let's go through the caused methods now. Generation of the first, accidental generation:

```
def generateRandom():
    result = []
    for i in range(0, generation_size * 2):
        genome = random.randint(0, 2147483647)
        result.append((genome, weight(genome)))
    return result
```

Here, and in the procedure of generating the new generation, we generate doubled number of individuals, that will participate in a tournament. Thus, we reject the most unadopted individuals ("freaks") who "are absolutely impractical". The following method is engaged in this initial rejection:

```
def sortGeneration(generation):
    if len(generation) > generation_size:
        generation.sort(key=lambda element: float(element[1]), reverse=True)
        generation = generation[:generation_size + 1]
    return generation
```



The above-mentioned procedure is able to sort a collection to genomes (are used rather short in record a lambda expression). As a result, the necessary quantity of the most well-adapted individuals is left.

Let's stop a little on the weightings procedure – fitness function calculations. It is carried out by the “Weight” method which as a parameter accepts a chain of genome bits:

```
def getx(genome):
    y = genome & 0xffff
    return y * (1.28 + 1.28) / (0x10000 - 1.0) - 1.28

def gety(genome):
    x = (genome >> 16) & 0xffff
    return x * (1.28 + 1.28) / (0x10000 - 1.0) - 1.28

def weight(genome):
    x = getx(genome)
    y = gety(genome)
    return 100.0 / (100.0 * (x ** 2 - y) ** 2 + (1 - x) ** 2 + 1)
```

Let's stop a little by the “GetX” and “GetY” methods. They take from a genome (32 bits) the part responsible for storage of parameters x and y, respectively, then they transform it into a floating comma format. At the same time the formula of transformation of representation from the previous section where  $\text{MIN} = -1.28$ ,  $\text{MAX} = 1.28$ ,  $N = 16$  (bit on parameter),  $2^N = 0 \times 10000$  is used.

And, at last, procedure of generating the new generation can take a following form:

```
def generateNewGeneration(parents, useElitism):
    result = []
    if useElitism:
        result.append(parents[0])
    while len(result) < generation_size * 2:
        parent1a = random.randint(0, generation_size)
        parent1b = random.randint(0, generation_size)
        parent2a = random.randint(0, generation_size)
        parent2b = random.randint(0, generation_size)
        parent1 = min(parent1a, parent1b)
        parent2 = max(parent2a, parent2b)
        mask = (~0 << random.randint(0, 32))
```

```

child = parents[parent1][0] & mask | parents[parent2][0] & ~mask

if random.random() > mutation_probability:
    child ^= 1 << random.randint(0, 32)

w = weight(child)

result.append((child, float(w)))
return result

```

The above mentioned method focuses on elitism selection, the tournament choice of ancestors, the crossover (single-point) and a mutation. Also, at generation of zero generation, this method generates doubled descendants, than it will be used for generating the new generation further. Our method (which is already considered above) “SortGeneration” will be engaged in this excess again. At those settings which are specified in the program the option called GA with elitism will be used. In our case, it is that one most adapted individual with a guarantee is introduced to new generation. Such an approach guarantees that the result will not be deteriorated from generation to generation, but often provides higher starting speed of decision search. At the same time, it can disturb more complete research of a decision space and accelerate degeneration of population. The reader can independently experiment with different settings and observe how they influence on the quality and the speed of finding solutions.

Having started the program with the given settings, it is quite often possible to receive here such result:

```

...
Generation 197
The best individual = 99,9999998156313
Generation 198
The best individual = 99,9999998156313
Generation 199
x = 0,999995727473869
y = 0,999995727473869
Genome = E3FFE3FF
Press "Input" for an exit...

```

The specified parameters x and y are the closest node to number 1.0, from a set of nodes which appear after splitting an interval  $[-1.28, 1.28]$  onto the 216 pieces. This result can be observed not always – we work with probabilistic (well, pseudo-probabilistic) process so we use a random number generator (class Random). And the decision space which needs to be investigated is rather extensive.

Of course, example is not the most difficult. It can be solved by different analytical and gradient methods. But if for you it is possible to bring parameters into a chain of bits and to code fitness function, and also arrange suboptimal value, then you can do without analysis of function on an extremum or that gradient method. If you do not have this knowledge, then such an approach will represent the costs connected with a challenge of the specialist or with independent deeper studying of subject domain. Of course, it is not free – at the expense of you as a specialist in programming and extremely irrational use of computer facilities. What is more, in each case you decide individually. But availability in your arsenal of such intellectual assistant as GA, will not disturb the precision.

#### **4.2.4. Closer to reality, or the space crossover**

Tasks similar to the solved in the previous section, a problem of searching an extremum of the test DeJong 2 function (see in De Jong 1975, De Jong and Spears 1991, De Jong and Spears 1992), are included in manuals on GA. It is clear why, the problem is foreseeable, the answer is known as well as easily checkable. At the same time, such examples leave dual impression. It is often easy to solve them also with others, simple methods – from accidental search (Monte Carlo method) to gradient methods such as coordinate optimization. On the contrary, they are difficult to solve by any methods.

The power of GA is revealed in examples in which the modeled system is divided into subsystems where solutions can be found in parallel. Often for overall performance for a specific objective, it is necessary to adapt classical genetic operators to invent new ones.

Other important point is that in real tasks the most labor-consuming part of process of the miscalculation is not a creation of a new generation or an assessment of fitness for each individual. Therefore some complications of the scheme of creating new generations slightly increase integral labor input of an algorithm, but also increase an “exit” of good descendants. Contrary to appearances, these complications greatly reduce the algorithm’s complexity.

Let’s try to show all this on the example of solving the task which is brought closer to real. So, the task is:

We are at war. Our opponent’s troops are unevenly located in a certain territory with weapons: 10 warheads of nuclear weapon. It is known that during the explosion of a warhead in blast radius, there is nothing left to live. Let’s just consider that beyond this radius, the enemy’s fighting units remain live. Our task is find such coordinates for each warhead that the opponent after the blow would have as few troops as possible.

Of course, at your more pacifistic moods, the same task is possible to said in other words:– there are funds for construction of 10 shops in a certain district of the city.

People are ready to go to the shop located not further than  $M$  meters from the house (by the analogy for the blast radius). People live unevenly on the chosen area. Your task is to arrange shops so as to cover the greatest number of inhabitants.

Further, in the program text, the names of identifiers are chosen from the first formulation of the task, but it will not mean specialization only for the first option of a specific problem.

The input data are images in the PNG format. This format is used because it can be squeezed without the loss of information, and it is important for determining a brightness of points. Fighting unit of the opponent is coded by a black point (brightness color a component in RGB – 0, 0, 0). We will show the territories covered with explosions in “tomato” color. Such a method associated with the submission of input data gives the chance to see the received result, and also to involve the graphic card of the computer in the assessment of the received result. It is possible just to draw the filled-in circles with a radius equal to the bomb blasting radius over the source image.

This time the program will present a WinForms-application with only one window as it is quite curious to watch dynamics of work of a genetic algorithm. And it is not really convenient to estimate the result only for the numbers running in the console in this case. Different application settings, as well as in the previous example, can be changed directly in the text of the program, using the IDE editor.

First of all, we will decide what option relating to coding the information within chromosomes we will use. The simplest option is to code coordinates in a sequential bit line as it was shown in the previous section. After that, it is possible to apply ordinary operators of the crossover and a mutation. But, practice shows that in such cases there are often different problems – a problem of the competing decisions, or a low quality of synthesizable decisions. Specifically, these problems are described in De Jong and Spears work (De Jong and Spears 1991). In the same paper also one of possible versions of solving these problems (the crossover based on space provision of decision points) is offered. Further in abbreviated form we will call it the space crossover.

So, we will store the data relating to one point in the form of not divided data set. In biology such signs are called linked because they are transferred to descendants, as a rule, together i.e., in our case it will be plane coordinates on X and Y axes.

In the program one “explosion” corresponds with the following class:

```
from random import random
from copy import copy, deepcopy
import math
from PIL import Image, ImageDraw

class Explosion:
    x = 0.0
    y = 0.0
```

```
def __init__(self, x: float, y: float):
    self.x = x
    self.y = y
```

```
@staticmethod
```

```
def generate_random(spec):
    return Explosion(random() * spec.bounds.width + spec.bounds.x,
                    random() * spec.bounds.height + spec.bounds.y)
```

Explosion coordinates are stored in this class (X, Y), and also this class provides several supplementary methods. The class “TaskSpecification” mentioned in one of methods above stores some restrictions and entry conditions of a task:

```
class Rectangle:
```

```
    x = 0
    y = 0
    width = 0
    height = 0
```

```
def __init__(self, x, y, width, height):
    self.x = x
    self.y = y
    self.width = width
    self.height = height
```

```
class TaskSpecification:
```

```
    bounds = Rectangle(0, 0, 0, 0)
    explosions_number = 0
    killing_radius = 0
    original_field = Image.new("RGB", (128, 128), (0, 0, 0))
```

```
def __init__(self):
    pass
```

```
def __init__(self, bounds: Rectangle, explosions_number: int, killing_radius: float,
            original_field: Image):
    self.bounds = bounds
    self.explosions_number = explosions_number
    self.killing_radius = killing_radius
    self.original_field = original_field
```

Above there are coordinates of borders beyond which it is not necessary to go at an arrangement of explosions (bound property), quantity of warheads (actually, it is genome length), defeat radius of each warhead and the original field of battle on which troops of the opponent are marked out and are provided sequent one after another.

Before giving a code of a class responsible for genetic information storage individual and working with it, we will look what means the space crossover.

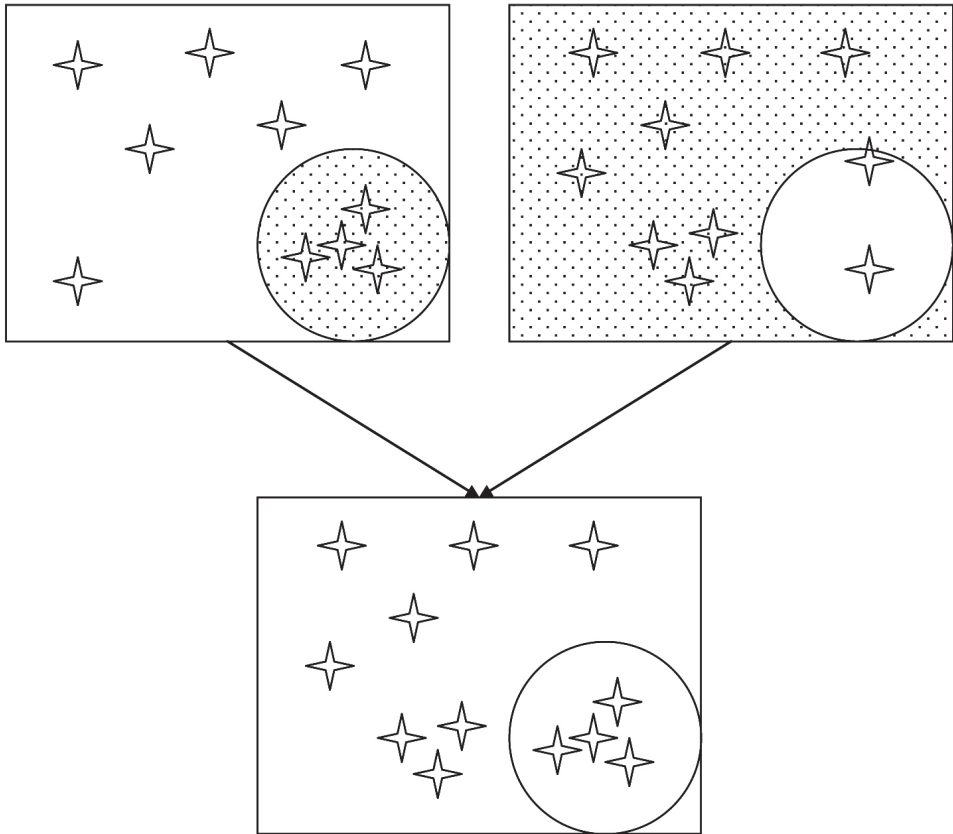
As it was already mentioned above if we get the list of spatial coordinates of two good, and different parents to apply regular one- or the two-point crossover, then, most likely, we receive the impractical descendant. Here are two basic reasons. First of all, at different individuals the same point can be stored in different places of a genome, and, at exchange of genetic information, we can place almost nearby geometrically two points which are in different places of a genome of parents. At the same time some other site will become bare. First, the genetic algorithm will spend all the power for selecting a successful method of coding coordinates, and, only then if, by this moment, there is no populations' degeneration yet, genetic algorithm (GA) will basically initiate the search of the successful decision. It is also a problem of the competing decisions – when it comes to the same decision there the set of methods relating to coding of this decision can be applied in a way which significantly increases the destructive property of the crossover.

The second problem is that if to pay attention only into place where the point in a genome is located, then the crossover will randomly take the points from one parent decision and from the decision of other. These points will be scattered on all search space. However it is clear that the points located close to each other often form peculiar sub complexes of decisions which support each other. And accidentally withdrawal or addition points in such a set of points significantly worsens the quality of decision.

Both of these problems are solved by the space crossover. During the work, it is guided by provision of points within the space of a solvable task. On the following sketch the scheme of consolidation of two parent genomes is shown below (see Fig. 4.1).

The space crossover consists of the following steps:

1. We choose any (accidental) point in space of a solvable task.
2. For both parents we cut out identical (on coordinates of the center and radius) circles (the sphere or a hypersphere for spaces with dimension more than two).
3. From one parent we take points which lie in a circle, from another – outside points.
4. We combine the taken points in one decision.
5. We adjust quantity of points in the descendant – we delete (in a random way) if they are superfluous, or we add the points that we have not used yet.



**Fig. 4.1.** The scheme of the space crossover

This approach allows to preserve more developed local ensembles of points more likely (in comparison with the ordinary crossover and method of coding).

Now, after short examination of the scheme referring to the space crossover, we can consider the chosen places of the class `Individual`. At the same time we will pass supplementary small methods not connected directly with GA to the place of a portrayal of images.

First of all, the genotype is stored in the list of points of explosions. There is also a field to which the fitness degree of an individual is brought. For us this is a number of black points that remained after explosion.

```
class Individual:
    fitness: float = math.inf
    explosions = List[Explosion]
```

```

def calc_fitness(self, spec: TaskSpecification):
    black_points = 0
    image = generate_bitmap(spec)
    pixels = image.load()
    for i in range(image.width):
        for j in range(image.height):
            if pixels[i, j] == (0, 0, 0):
                black_points += 1
            pass
    fitness = black_points

```

```

def generate_bitmap(self, spec: TaskSpecification):
    result = deepcopy(spec.original_field)
    draw = ImageDraw.Draw(result)
    for explosion in explosions:
        draw.ellipse(explosion.x - spec.killing_radius, explosion.y - spec.killing_radius,
                    spec.killing_radius * 2, spec.killing_radius * 2, fill="red", outline="red")
    draw.flush()
    return result

```

The first generation (see more in Diaz-Gomez and Hougen 2007) is created in a random way by means of the static “generate\_random” method as follows:

```

@staticmethod
def generate_random(self, spec: TaskSpecification):
    result = Individual()
    for i in range(0, spec.expositions_number):
        result.append(Explosion.generate_random(spec))
    pass

```

Now we will consider implementation of genetic operators. The mutation is simple. Any point of the decision is replaced in a random way. For our case it is sufficient, but for a large number of points (hundred, thousands of explosions) it may be necessary to extend the implementation so that it can place more points at once.

```

def mutate(self, spec: TaskSpecification):
    explosions[random.randrange(len(explosions))] = Explosion.generate_random(spec)

```



The code of the space crossover is longer. It can takes the following form:

```
def spatial_crossover_with(self, parent2: Individual, spec: TaskSpecification):
    x0 = random() * spec.bounds.width + spec.bounds.width
    y0 = random() * spec.bounds.height + spec.bounds.height
    radius = random() * math.max(spec.bounds.width, spec.bounds.height)
    new_genome = List[Explosion]
    used1 = List(spec.expositions_number)
    used2 = List(spec.expositions_number)
    deleted = List(spec.expositions_number * 2)
    for i in range(spec.expositions_number):
        sp = explosions[i]
        if radius > math.abs(sp.x - x0) + math.abs(sp.y - y0):
            new_genome.append(sp)
            used1[i] = True
    for i in range(spec.expositions_number):
        sp = parent2.expositions[i]
        if radius > math.abs(sp.x - x0) + math.abs(sp.y - y0):
            new_genome.append(sp)
            used2[i] = True
    if len(new_genome) > spec.expositions_number:
        num = len(new_genome) - spec.expositions_number
        for i in range(num):
            del_num = 0
            while True:
                del_num = random.randrange(len(new_genome))
                if not deleted[del_num]:
                    break
            deleted[del_num] = True
        tmp_genome = List[Explosion](spec.expositions_number)
        for i in range(len(new_genome)):
            if not deleted[i]:
                tmp_genome.append(new_genome[i])
        new_genome = tmp_genome
    elif len(new_genome) < spec.expositions_number:
        while len(new_genome) < spec.expositions_number:
            add_num = 0
            while True:
```

```

    add_num = random.randrange(spec.expositions_number)
    if not used1[add_num]:
        break
    used1[add_num] = True
    new_genome.append(explosions[add_num])
explosions = new_genome

```

Here we see all those steps which were described for the space crossover. But there is also a feature. At addition of points from ancestors, we measure distance to a point in  $L_1$  space (the Cityblock metric). It can be noticed, having analyzed a fragment “radius > math.abs(sp.x – x0) + math.abs(sp.y – y0)”. Essentially it changes nothing, just the squares (turned by 45 degrees), but not circles will be cut out from space of parents.

Also, during the initial testing of an example, it turned out that after initial progress and finding the most perspective configurations of decisions, there was an extremely slow process associated with the adaptation of received decisions, generally at the expense of a mutation. But the mutation in the form, in which it was stated above, is a rather rough medium – instead of one explosion we receive another one randomly. And the probability of a slight correction of the previous one is very small. Therefore, one more operator which is called Stepping was added to this example. Actually it is also a mutation, but a soft one which slightly displaces an explosion to which it is applied:

```

def limit(self, value, min_value, max_value):
    if value > max_value:
        return max_value
    if value < min_value:
        return min_value
    return value

def stepping(self, spec: TaskSpecification):
    index_of_change = random.randrange(len(explosions))
    shift_x = (random() – 0.5) * spec.killing_radius * 0.25;
    shift_y = (random() – 0.5) * spec.killing_radius * 0.25;
    explosion = explosions[indexOfChange];
    explosion.X = limit(explosion.x + shift_x, spec.bounds.x, spec.bounds.x + spec.
        bounds.width)
    explosion.Y = limit(explosion.y + shift_y, spec.bounds.y, spec.bounds.y + spec.
        bounds.height)

```

Let's take a look at what turned out. The source image (it is possible to find in the list of initial files of an example under a name  $128 \times 128\_2.png$ ) has the size of  $128 \times 128$  pixels. Initially on the image of 5910 black pixels which we also should “cover with explosions” as much as possible.



Fig. 4.2. Initial area for defeat radius

Radius of defeat area – 12 pixels, the number of explosions – 10. The population size – 1000 individuals. The elitism, tournament selection for 500 best individuals was used. Probability of use of the space crossover – 0.4, mutations – 0.1, a stepping – 0.3. Of course, at each start results will differ a little – we deal with probabilistic process.

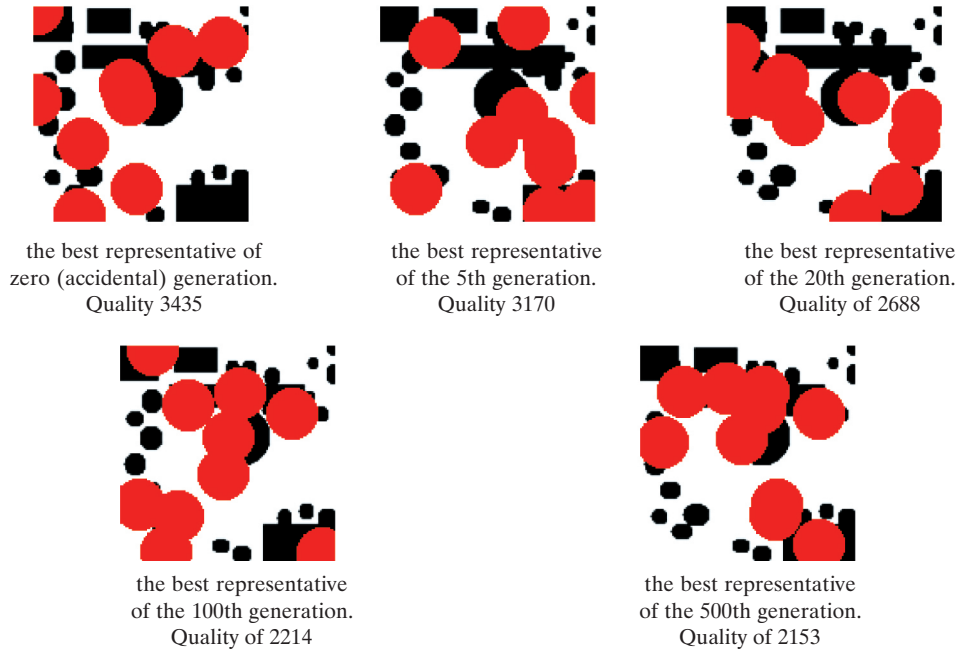


Fig. 4.3. Results of applying an evolutionary algorithm

Work with the program happens as follows – we start application, we load the necessary picture (“Loadsample” button), we start evolution (we press the Start button). The current result of evolution is reflected in a main window of the program, and intermediate results are registered in the same catalog from which the picture was loaded (the file with the same name and the TXT expansion, and also png-pictures).

In the given example only one of possible adaptations of GA to practice was shown. The nature is only an initial role model, but we are not obliged to copy all details. It is enough to observe only the basic principles and use common sense. Let’s enlist (in Tab. 4.1) some principles allowing to achieve a faster convergence of evolution process or to improve quality of a subject domain research.

**Tabela 4.1**

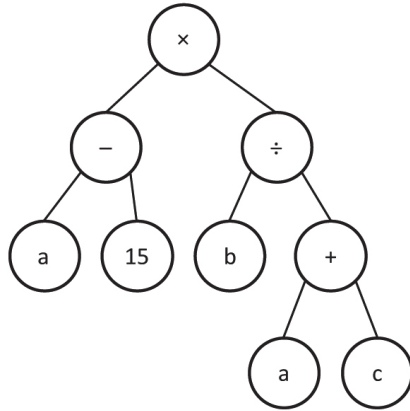
The list of principles improving the algorithm convergence

<b>Acceleration of convergence of the decision</b>	<b>Improvement of the work quality (search capabilities) of GA</b>
<ul style="list-style-type: none"> <li>– Increase in pressure of natural selection               <ul style="list-style-type: none"> <li>• Reduction of quantity of the individuals allowed to reproduction</li> <li>• Use of elitism</li> <li>• Reduction of total quantity of the generated posterity</li> </ul> </li> <li>– Algorithm execution in parallel on several computers (processors)</li> </ul>	<ul style="list-style-type: none"> <li>– Reduction of pressure of natural selection</li> <li>– Increase in volume of genetic material</li> <li>– Diploid GA (also increases amount of genetic material, see in Bara’a 2004)</li> <li>– Splitting population into parts. It also is method easy to parallelize an algorithm</li> </ul>

#### 4.2.5. Genetic programming

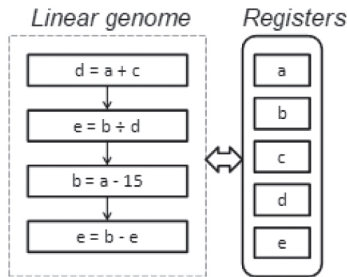
The above mentioned genetic algorithm, works with information coded in genes. What information, in general, is not important? For example these might be the processor commands of a computer. The program and the quality of the received individual will depend on how efficiently this program comes to an objective. The main minus will be the fact that length of the program will be fixed. And classical genetic operators do not promote high probability of the appearance of efficient descendants.

Therefore other algorithms have been developed for evolutionary programs developing (in comparison with GA) genome storage methods and other implementations of genetic operators. Chronologically, the tree form of a genome storage was the first form (Fig. 4.4), offered by N. Cramer (Cramer 1995, Koza 1989, Koza 1992, 1994).

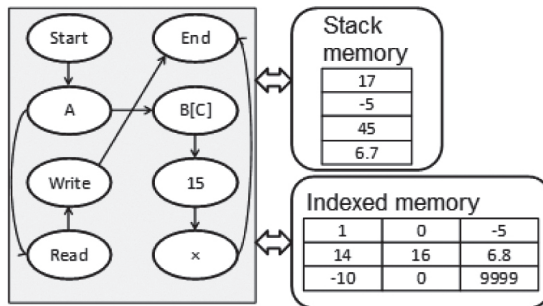


**Fig. 4.4.** The algorithm of calculation of value of function  $(a-15)(b/(a+c))$  presented in the tree form

Apart from other forms, which are also often used, we can distinguish linear (Fig. 4.5) as well as network (or graph) forms (Fig. 4.6).



**Fig. 4.5.** The algorithm of value of function  $(a-15)(b/(a+c))$  presented in the linear form (a code of some virtual processor)



**Fig. 4.6.** The genome of the small program for the virtual processor presented in the network (graph) form

Transaction of a mutation, in case of the genetic programming (GP), is similar to close transaction at GA, but a variety of changes which can be brought into a genome is added. For a linear genome, for example, to random change of any team, transactions relating to an insert of accidental team or removal in the accidental place are added.

A bit more difficult is also a transaction of exchanging genetic material in ancestors – the crossover looks as if the basic principle remained the same and was a part of the genome of one parent and a part of the other. Let’s give examples of the crossover for linear (Fig. 4.7) and treelike (Fig. 4.8) representation of a genome.

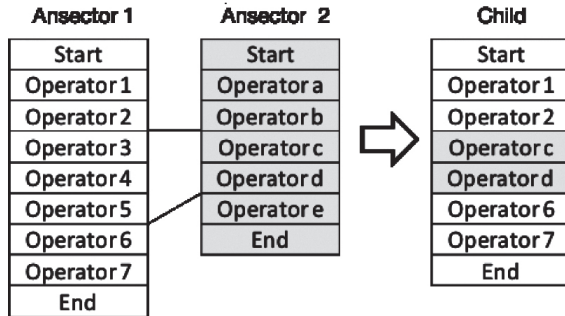


Fig. 4.7. The crossover for the linear representation of a genome

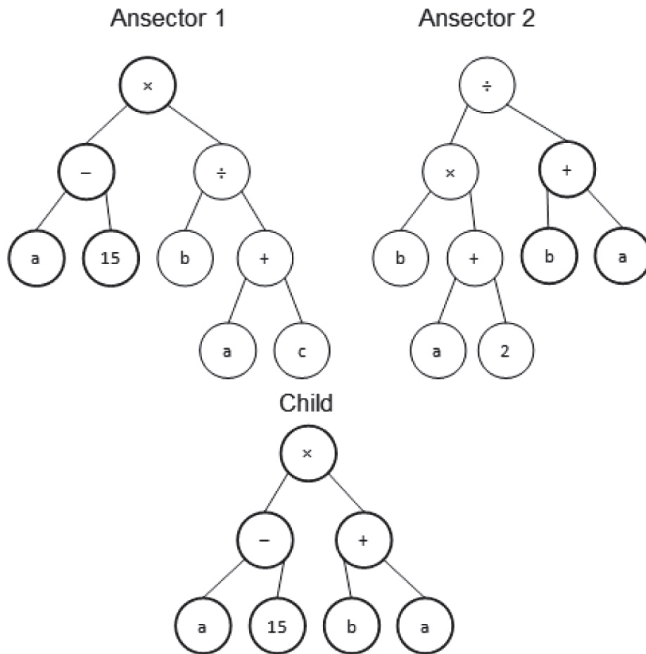


Fig. 4.8. The crossover at the treelike representation of a genome

Transaction of the crossover is extremely destructive for usual programs and the overwhelming number of descendants after such transaction become impractical. Therefore, researches often enter unusual transactions into system of teams, for instance, transition on complementary tags (see for instance De Jong and Spears 1991, 1992), the automatically defined functions (ADF) (see for example Ferriera 2006, Federicks and Chen 2013) and a lot of other improvements.

Programs also “fight” against what destroys them. For example, they increase quantity of introns – pieces of a code which contain nothing related to the transactions (see in Bentley and Wakefield 1996, Goldberg and Sastry 2001):

```
(NOT (NOT X))
(AND... (OR X X))
(+... (-X X))
(+ X 0)
(* X 1)
(*... (DIV X X))
(MOVE-LEFT MOVE-RIGHT)
(IF (2=1)... X)
(A: = A)
```

Such method of protection (as well as duplication of the most important sites of a genome) is not the invention of GP. Similar mechanisms are available also in genomes of usual cages. All subtleties of genetic programming can be examined as a subject of the separate book (and not the one). Therefore we will recommend the interested reader continuing this study on more specialized literature. Good introduction are works: Bentley and Wakefield 1996, Goldberg 1989, Goldberg and Sastry 2001, Michelle 1996, Whitley 1994, Whitley 2001. Specifically we will review rather simple example of the search for the decision related to a genome of variable length.

#### **4.2.6. To be, or not to be...**

As it was told in the famous Soviet movie – “friends, but whether not to threaten to us on William of our Shakespeare?” Also we will try to perform onstage using immortal monologue “to be or not to be”. And method which we intend to implement could be certainly used by Johnathan Swift<sup>1</sup> as well.

---

<sup>1</sup> The work “Gulliver’s Travel” where the inventor who constructed the car generate accidental combinations of all existing words and describe their meanings. Those intelligent sets were registered to create the complete encyclopedia of all sciences and arts. Of course, Swift in this case criticized this direction – but he then knew nothing about genetic programming.

Let's consider the following task. We need to guess some phrases. To find out, if the algorithm guessed the phrase or not, we take any line, and determinate a number (fitness function) which represents the difference the number of letters standing on their places, with those that do not take their position:

$$f(a, b) = \sum_{i=0}^{\max(\text{length}(a), \text{length}(b))-1} c(a, b, i),$$

$$c(a, b, i) = \begin{cases} 1, & i < \text{length}(a) \cap i < \text{length}(b) \cap a[i] = b[i] \\ -1, & \text{else} \end{cases}.$$

Here “a” stands for line sample. It is known only to the calculator of fitness function. “b” – a line which we showed for assessment; length() – the function calculating length of the transferred line. An indexation of symbols begins with a zero position.

Having only such scanty data and the approximate size of the text, we will try to guess the phrase from the immortal work. Namely, Hamlet's monologue *to be, or not to be* can be taken into account. The text of the monologue is taken from the [https://en.wikipedia.org/wiki/To\\_be,\\_or\\_not\\_to\\_be](https://en.wikipedia.org/wiki/To_be,_or_not_to_be) page, with all punctuation marks.

The appendix “SimpleGP” is an object oriented application. All data and settings are hardcoded therefore if we intent to change them, it is necessary to use IDE and to recompile the program. Let's start with the point of entry in the program – the “main.py” file:

```
def print_world_state(generation, world):
    print("Best individual ", generation, ":\n")
    print(" Genome=", world.generation[0].genome, "\n")
    print(" Length=", len(world.generation[0].genome), "\n")
    print(" Fitness=", world.generation[0].fitness, "\n")
    print("-----\n")

if __name__ == "__main__":
    world = GPWorld()
    generation = 0
    while world.generation[0].genome != GPWorld.SECRET_PATTERN:
        print_world_state(generation, world)
        world.next(True)
        generation += 1
    print_world_state("last", world)
    print("Solution is found!")
```



The “\_\_main\_\_” method includes generation of zero generation and a basis cycle – generation of new generations. The cycle is interrupted when the exact phrase is found – it is stored in the class “GPWorld” constant “SECRET\_PATTERN”. But this storage location is chosen only for simplification of an example. Basically, the source text can be stored anywhere on other computer. The main thing is that the program has two options to execute: comparison (as condition of the termination of a cycle) and similarity assessment (fitness-function calculation).

The “WriteWorldState” method brings the best representative of the current population to the console.

In the class “Program” two more classes are used: the class “Individual” (encapsulates an individual) and the class “GPWorld” (encapsulates population and transactions over it and separate individuals). Let’s consider them in more detail. “Individual.py” can take a following form:

```
class Individual:
    genome = ""
    fitness = 0

    def __init__(self, genome: str, fitness: int):
        self.genome = genome
        self.fitness = fitness
```

The above mentioned class is simple. In the fields of the class, we store only a genome (line) and the cached value of fitness-function (generally for sorting individuals within the population). The main logic of genetic transactions is concentrated in the “GPWorld.py” file.

```
SECRET_PATTERN = ""To be, or not to be, that is the question;
...
And lose the name of action.""
SECRET_PATTERN_LENGTH = len(SECRET_PATTERN)
```

Here we are going to reduce a multi lowercase line literal. It can be seen completed in the initial files attached to the book. Also, it is possible just to start the program and to wait for its termination.

```
generation = []
generation_size = 100000
crossover_probability = 0.4
mutation_probability = 0.2
```

This block of constants stands for setup relating to parameters of an example.

```
def __init__(self):
    self.generation = GPWorld.generate_random_generation()
    self.generation.sort(key=lambda element: element.fitness, reverse=True)
```

The designer generates new accidental population. Next they are sorted by fitness-function in decreasing order.

```
def generate_random_generation():
    result = []
    print("Generating zero generation...\n")
    for i in range(0, GPWorld.generation_size):
        genome = ""
        length = random.randint(1, GPWorld.SECRET_PATTERN_LENGTH * 2)
        for l in range(0, length):
            genome += GPWorld.random_possible_char()
        ind = Individual(genome, GPWorld.calc_fitness(genome))
        result.append(ind)
        print("\r", i)
    print("\n")
    return result
```

The supplementary method “random\_possible\_char()”, as its name suggests, returns accidentally chosen symbol from all possible.

```
def random_possible_char():
    chars = "abcdefghijklmnopqrstuvwxyz' ABCDEFGHIJKLMNOPQRSTUVWXYZ,;.-? \\\n"
    return chars[random.randrange(0, len(chars))]
```

The variable chars sets the list of possible symbols which can occur in the required phrase. In this case we were limited to the English alphabet, punctuation marks, as well as a gap and line feed.

The “calc\_fitness()” method calculates fitness-function according to the verbal description set at the beginning of the section.

```
def calc_fitness(genome: str) -> int:
    result = 0
    genome_length = len(genome)
```

```

max_length = max(GPWorld.SECRET_PATTERN_LENGTH, genome_length)
min_length = min(GPWorld.SECRET_PATTERN_LENGTH, genome_length)
for i in range(0, max_length):
    if i >= min_length or genome[i] != GPWorld.SECRET_PATTERN[i]:
        result -= 1
    else:
        result += 1
return result

```

The “next()” method generates a new generation. The accepted parameter of Boolean type specifies whether the strategy of elitism is used. If it is used – one best copy of the previous generation with guarantee passes into the following without changes. The selection is made by a tournament method for representatives of the most adapted half of the population. Thus, selection pressure amplifies a little.

```

def next(self, use_elitism: bool):
    new_generation = []
    if use_elitism:
        new_generation.append(self.generation[0])
    while len(new_generation) < GPWorld.generation_size:
        parent1a = random.randrange(0, GPWorld.generation_size // 2)
        parent1b = random.randrange(0, GPWorld.generation_size // 2)
        parent2a = random.randrange(0, GPWorld.generation_size // 2)
        parent2b = random.randrange(0, GPWorld.generation_size // 2)
        parent1 = min(parent1a, parent1b)
        parent2 = max(parent2a, parent2b)
        if random.random() < GPWorld.crossover_probability:
            new_genome = GPWorld.crossover(self.generation[parent1].genome,
            self.generation[parent2].genome)
        else:
            new_genome = self.generation[parent1].genome
        if random.random() < GPWorld.mutation_probability:
            new_genome = GPWorld.mutate(new_genome)
        ind = Individual(new_genome, GPWorld.calc_fitness(new_genome))
        new_generation.append(ind)
    new_generation.sort(key=lambda element: element.fitness, reverse=True)
    self.generation = list(new_generation)

```

The “mutate()” method with an equal probability carries out transactions of an insert of an accidental symbol, removal or replacement.

```
def mutate(genome: str):
    if random.random() < 0.33:
        pos = random.randint(0, len(genome))
        genome = genome[0:pos] + genome[pos:]
    if len(genome) > 0 and random.random() < 0.33:
        pos = random.randrange(0, len(genome))
        genome = genome[:pos] + genome[pos + 1:]
    if len(genome) > 0 and random.random() < 0.33:
        pos = random.randrange(0, len(genome))
        new_genome = ""
        for i in range(0, len(genome)):
            if i != pos:
                new_genome += genome[i]
            else:
                new_genome += GPWorld.random_possible_char()
        genome = new_genome
    return genome
```

Transaction of the crossover is similar to the single-point crossover at usual GA, but is adapted for processing of genomes of variable and unequal length. The resulting genome has length equal to genome1. If the second genome is shorter, then accidental symbols from a set of admissible are added to a line. If it is longer – than only a part of symbols within length of the first genome is used.

```
def crossover(genome1: str, genome2: str):
    result = ""
    pos = random.randrange(0, len(genome1))
    for i in range(0, len(genome1)):
        if i > pos:
            result += genome1[i]
        elif i < len(genome2):
            result += genome2[i]
        else:
            result += GPWorld.random_possible_char()
    return result
```

In this case of the crossover there are no transactions which lead to shift of line fragments either at the beginning or at the end. It is connected with the fact that estimated function estimates only the symbols which precisely got to the position. Therefore any shifts bring the fact that in each case genome is damaged. Similar transactions (as it is shown in Fig. 4.4), are possible to be implemented independently together with fitness-function modification. In this case fitness-function should add points to a genome if it includes sublines, even if they do not stand on the places.

So, now we are ready for start. We start the program watching algorithm's work. The following results are received of this program:

Generation 0:

```
Genome=EA-B; - gOcLaRGGhjaQonoAFaHWz
S? QIBBQs; UaoUlpGBUJYLoUeU
pd -.eEh? g.XqfYWpoPaLm
...
; u, sW.rAFbkYs
Length=1409
Fitness=-1350
```

Generation 500:

```
Genome=TE be, or notGto b?, tpan.' Ghe eBeJtiln;
WhetRer Vpis irblgrBgT tfekqi dxto iuWdlrw
The SliJ? sLaln Ao
...
ADd losG theonCme of a-tikn.
Length=1442
Fitness=182
```

Generation of 1000:

```
Genome=To be, or not to be, that is the qBestion;
Whether 'tis nobler in the mind to sufder
The Slings ann Armwms of outrageous
...
ADd lose the name of action.
Length=1442
Fitness=1066
```

Generation of 1426:

Genome=To be, or not to be, that is the question;  
Whether 'tis nobler in the mind to suffer  
The Slings and Arrows of outrageous Fortune

...

And lose the name of action.

Length=1442

Fitness=1442

#### 4.2.7. Diophantine equation

In the last paragraph, dedicated to genetic algorithms, we would like to present an example of solving a Diophantine equation using, probably, the most powerful and true Python library – DEAP. DEAP is an innovative evolutionary computing platform for rapid prototyping and testing of ideas. It seeks to make algorithms explicit and data structures transparent. It works in perfect harmony with parallelization mechanisms such as multiprocessing and SCOOP.

In mathematics, a Diophantine equation is a polynomial equation, usually with two or more unknowns, such that only the integer solutions are sought or studied (an integer solution is such that all the unknowns take integer values). A linear Diophantine equation equates the sum of two or more monomials, each of degree 1 in one of the variables, to a constant. An exponential Diophantine equation is one in which exponents can be unknowns.

As the example, we take a linear Diophantine equation:  $1027 \cdot x + 712 \cdot y = 1$ .

The example of usage of Genetic Algorithms and DEAP is shown on the listing below, and code is aimed to show how a task can be solved with Genetic Algorithms. Sure thing, that the problem has the analytical solution.

```
import random
import operator

from deap import tools, base, creator, algorithms

MIN, MAX = -30, 30
VARIABLES = 4

MUT_MIN, MUT_MAX = -10, 10
NGEN, IND_SIZE, CXPB, MUTPB, TRN_SIZE = 10000, 50, 0.5, 0.5, 1000
HALL_SIZE = 10
DEFAULT_MAIN_ARGS = NGEN, IND_SIZE, CXPB, MUTPB
```

```

BEST_INSTANCE_MSG = 'Best instance =>'
NO_SOLUTION_MSG = 'No solution found in integers. Distance =>'

def f(a, b, c, d):
    return abs(a + 2*b + 3*c + 4*d - 30)

def fitness(instance):
    a, b, c, d = instance
    return f(a, b, c, d),

def spawn_instance():
    return random.randint(MIN, MAX), random.randint(MIN, MAX)

def mutate(instance, mutpb):
    if random.random() <= mutpb:
        index = random.randint(0, len(instance) - 1)
        instance[index] += random.randint(MUT_MIN, MUT_MAX)
    return instance,

def get_best_result(population):
    if isinstance(population[0], list):
        fitness_values = list(map(fitness, population))
        index = fitness_values.index(min(fitness_values))
        return population[index]
    else:
        return min(population, key=operator.attrgetter('fitness'))

def terminate(population):
    if fitness(get_best_result(population)) == (0, ):
        raise StopIteration
    return False

def distance_from_best_result(population):
    result = get_best_result(population)
    return fitness(result)[0]

def output(best_instance):
    print(BEST_INSTANCE_MSG, best_instance)
    distance = fitness(best_instance)

```

```

if distance == None: return
if len(distance) > 0 and distance[0] != 0:
    print(NO_SOLUTION_MSG, distance)
pass

def setup(mutpb):
    creator.create("FitnessMin", base.Fitness, weights=(-1,))
    creator.create("Individual", list, fitness=creator.FitnessMin)
    toolbox = base.Toolbox()
    toolbox.register("attribute", random.randint, MIN, MAX)
    toolbox.register("individual", tools.initRepeat, creator.Individual,
                    toolbox.attribute, n=VARIABLES)
    toolbox.register("population", tools.initRepeat, list, toolbox.individual)
    toolbox.register("mate", tools.cxOnePoint)
    toolbox.register("mutate", mutate, mutpb=mutpb)
    toolbox.register("select", tools.selBest)
    toolbox.register("evaluate", fitness)
    return toolbox

def main(ngen, ind_size, cxpb, mutpb):
    toolbox = setup(ind_size)
    population = toolbox.population(n=ind_size)
    stats = tools.Statistics()
    stats.register("best_instance_of_population", get_best_result)
    stats.register("distance", distance_from_best_result)
    stats.register("terminate", terminate)
    halloffame = tools.HallOfFame(HALL_SIZE)
    try:
        algorithms.eaSimple(population, toolbox, cxpb, mutpb, ngen,
                           stats=stats, halloffame=halloffame)
    except StoptIteration:
        pass
    finally:
        best_instance = halloffame[0]
        output(best_instance)
        print("Checking answer... f(a, b, c, d) => " \
              + str(f(best_instance[0], best_instance[1], best_instance[2], best_instance[3])))

if __name__ == '__main__':
    main(*DEFAULT_MAIN_ARGS)

```



### 4.3. Swarm intelligence

Many techniques relating to soft computing are based on the idea of representing a set of rather simple objects combined by quite clear rules. Specifically, these techniques show the behavior going beyond the activity of a separate individual on average intellectual level. Here it is possible to apply either neural networks or evolutionary algorithms. This paper will not examine neural networks in detail as it is necessary to bring up this topic in completely separate course. Fortunately, the theory of neural networks is worked and described rather well for the reader of any level therefore there is a desire to simply find such manual which can allow to penetrate this subject more deeply. We got acquainted with evolutionary algorithms in the previous section. But probably the most brightly and obviously the idea related to summing of individual intelligence, is implemented in such a direction of research as Swarm Intelligence.

With the development of computer architecture for the nearest future, a number of dominant trends were determined: the number of cores in the processors will grow, and individual computers will be combined into various kinds of network infrastructures, clusters and grids. In other words, there is something like a transition from unicellular organisms to multicellular organisms or from a single existence to a communal one that occurred in nature more than 500 million years ago. With the coming communality in IT, new challenges also arise: one has to learn how to manage a variety of distributed infrastructures, find ways to integrate millions of cores and distribute the load between them, and so on.

Now in the IT field, the engineering approach to the decision of such problems prevails, but there are also many others – in particular, the once built on the principles of bionics by combining engineering and biological principles. In bionic algorithms, prompted by nature (Nature Inspired Algorithms, NIA), the behavior of colonies of insects, birds or fish is reproduced on the machine level. NIA is not new – they have found application in various kinds of optimization applications, and recently they have extended it to management (see for example Bonabeau et al. 1999, Dorigo 1992, Gambardella and Dorigo 2000, Goss et al. 1989, Kennedy 2010, Negnevitsky 2005, Smaldon and Freitas 2006). And if you take into account that natural processes have natural parallelism, NIA performance can be significantly accelerated by using large pools of graphics processors and technologies like MapReduce.

The most NIA solutions imitate the properties of communities consisting of primitive individuals – more precisely, their amazing ability to make decentralized decisions. Communities of simple organisms provide a vivid example of the synergistic effect, when the aggregate capacity of a community is greater than the sum of the capabilities of individual parts – from which collective components, collective intelligence

or swarm intelligence (SI) are formed. Thanks to SI, the swarm demonstrates the behavior and acceptance of such solutions, which, because of their complexity, are certainly unavailable to a single individual. Who did not observe the behavior of ants or unusual pirouettes of avian packs?

More recently, in the early 1990s, Deborah Gordon (Gordon 1999), a researcher at Stanford University, uncovered the underlying secrets of coherent behavior. She studied the mechanism SI on the example of the activity of termite ants, each of which does not possess any intelligence, but their colony as a whole acts very reasonably. It turned out that SI is a product of some structural complexity and is formed due to a system of simple communications between members of the community. Exchange by several bytes was enough to organize an expedient decentralized collective behavior. The numerous studies that followed at the end of the last century showed that the collective behavior of bee colonies, avian schools and even human communities is based on the same principles. The latter was well written by James Surowiecki (Surowiecki 2005) in the book “The Wisdom of the Crowd”, which explains the phenomenon of crowdsourcing.

The discovery of Gordon was the impetus for a large number of research and development, breathed life into an almost forgotten bionics. The first was the well-known problem of the traveling salesman, followed by many others, described in detail, but when considering SI from a variety of facts, we confine ourselves here only to the fact that it involves the design of intelligent multi-agent systems based on principles borrowed from the collective behavior of insect communities. How can the model of coordinated behavior of the colony, based on simple interactions between an individual member and the colony as a whole, be used for applied purposes?

The first practical applications of SI for optimization applications date back to the mid-nineties of the last century, and in the second decade of the present century, opportunities were opened for the extension of SI methods to infrastructures such as clusters or grids. The huge potential of NIA here is obvious – there is a direct analogy between the set of somehow united processors and colonies of animals. Another area of possible application of SI in IT is the Big Data problem, where NIA-based technologies can support distributed computing models of MapReduce. The limitations of existing implementations of MapReduce in that they use a specific number of cores, hence the inevitability of the batch mode and the structuring of the output data. But if the idea of MapReduce implemented a swarm consisting of several types of nodes that can cope with any input data combined with the ability to self-organize, then such a swarm could quite work in real time, and without any preliminary data organization. Intelligence opens up new possibilities in data analysis fundamentally, but this approach does not apply to transactional systems, where ACID requirements are critical (Atomicity, Consistency, Isolation, Durability).

Why does this direction involve researchers so much? Probably, because in the surrounding world, all of us see examples of such “intelligence”:

- Animals (the person as the highest manifestation) – it is already clear that the system rather simple is responsible for all very difficult behavior (but still not until the end of studied) elements, neurons.
- Groups of scientists are most often able to solve more difficult problems in the intellectual relation, than the singles. Though, very often there is not parallel consolidation, but consecutive – schools of sciences, the teacher pupils.
- Production of computers (and any difficult modern technique) – we will risk to claim that any of recent people does not know a cycle of production of the computer in all details. We can say the same about oil production for production of plastic or metal for production of wires, finishing with questions about design of advertising booklets for shops. Nevertheless, many people successfully make and sell all this equipment.
- Ants and bees – a swarm of bees or ant colony for the dependent onlooker make an impression of something much more intellectual than separate individuals. For example we can observe the process of searching for food and the optimization of finding the way on which ants bring found food in their ant hill.

The last example is modeled in “Ant algorithms” from which we want to start studying the direction of Swarm Intelligence.

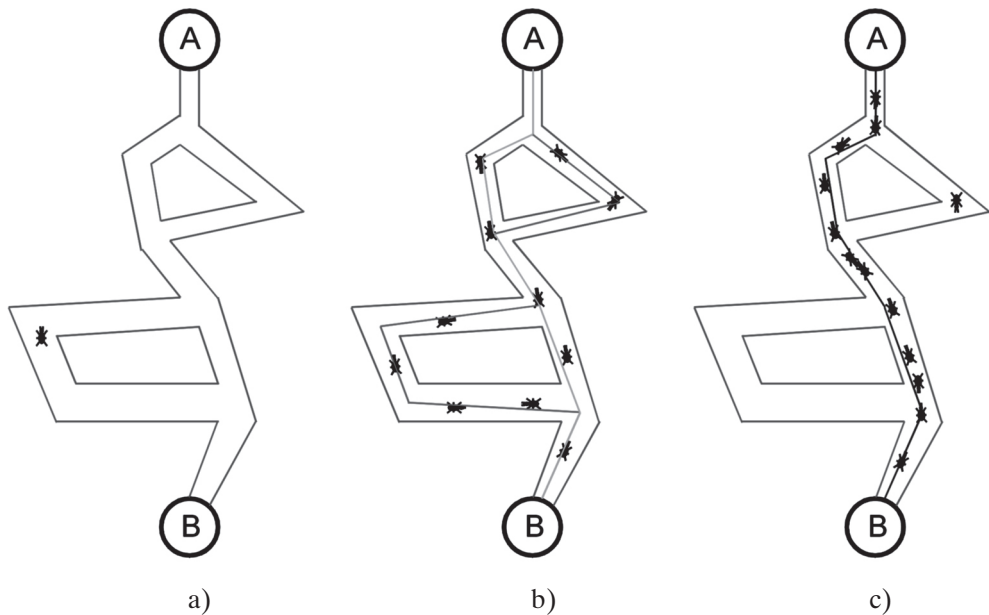
The original algorithm (Ant Colony Optimization) arose while observing how Argentine Ant live inspect the territory around an ant hill, find food and bring it to an ant hill, constantly optimizing (reducing) the path each of ants goes through. These researches were examined in 1989 by Goss and et al. (Goss and et al. 1989) and in 1993 by Denebourg et al. (Denebourg et al. 1993). The first mathematical formalization of an algorithm was offered in 1992 by Marko Dorigo (Dorigo 1992, Dorigo and Stützle 2004) and Gordon (Gordon 1999).

Ants while searching for food go around an ant hill in a random way through tracks which were not previously bitten (“trodden”) by generations of insects. The point is that the main orientation is based on giving off pheromones to which they are very sensitive and with which they mark everything around. Moreover, each ant hill has the individual train pheromone. Thus, the ant which has the same appearance but comes from separate ant hill, will be apprehended as the enemy. Among ants we can also distinguish blind ants which are guided by trail pheromones as well as touch.

Having found food, the ant intelligence agent comes back home, using one of tracks. At the same time all its way is marked with a special pheromone. Upon return to an ant hill, this ant “calls” (we will call this action so that not to complicate the description) other working ants and the process of transferring food in an ant hill begins.

Working ants are guided by the smell left by an ant intelligence agent. At the same time, they strengthen a smell, leaving the marks on a path if they find food. Thus, footpaths become more and more noticeable. But they not always go the same way – sometimes ants go astray and then in a random path look for the place marked with a pheromone. Sometimes it leads to finding a shorter way. Also, at this time there is an evaporation process of pheromones. If it didn't occur, then the initial way would always have the strongest smell and process of searching for a shorter way would not happen.

Let's provide this process graphically on Figure 4.9.



**Fig. 4.9.** An optimization process of transport path of the found food (A) to an ant hill (B):  
a) ant finds food and comes back home any way; b) working ants convey food to home following and laying pheromones; c) the most part of ants moves on the shortest way

In Figure 4.9:

- a) an ant as the intelligence agent finds food then comes back home any way;
- b) working ants convey food to an ant hill following and laying pheromones trails; on shorter way ants manage to pass in bigger quantities therefore gradually this way is becoming more and more intense in terms of pheromones;
- c) the most part of ants moves on the shortest way and only separate individuals use other ways.

Let's describe the elementary ant algorithm more precisely. The labyrinth is set by the count (tops and edges). We consider the ants looking for an optimum way (shortest) between two tops (an ant hill and food).

So far (exit conditions are not satisfied)

1. We create ants.
2. Ants look for decisions.
3. Updating of level of a pheromone.

Let's consider each of cycle steps in more detail.

1. Starting points where ants are located, depend on restrictions of a task. In the elementary cases, we can place all of them in one point, or distribute accidentally on the area of a labyrinth. At the same stage each edge of a labyrinth is marked with the small positive number characterizing a pheromone trail. It is necessary in order that on the following step we had no zero probabilities (see in Denebourg et. al. 1993).
2. We determine the transition probability from  $i$  top in  $j$  top by the following formula:

$$P_{i,j}(t) = \frac{\tau_{i,j}(t)^\alpha (d_{i,j})^{-\beta}}{\sum_{j \in \text{connected nodes}} \tau_{i,j}(t)^\alpha (d_{i,j})^{-\beta}}.$$

Where  $\tau_{i,j}(t)$  – level of a pheromone trail,  $d_{i,j}$  – heuristic distance,  $\alpha, \beta$  – constants.

If  $\alpha = 0$ , then the choice of the closest neighbor is the most probable, and the algorithm becomes “greedy”.

In a case  $\beta = 0$ , the choice only on the basis of pheromone level is the most probable so it leads to jamming on already “trodden” ways.

As a rule, some compromise value is used for these sizes chosen experimentally for each task.

Updating of level of a pheromone is made as follows:

$$\tau_{i,j}(t+1) = (1-\rho)\tau_{i,j}(t) + \sum_{k \in \text{used edge}(i,j)} \frac{Q}{L_k(t)}.$$

Here  $\rho$  – the parameter setting intensity of evaporation  $L_k(t)$  – the price of the current decision for  $k$ -th of an ant, and  $Q$  stands for a price order for the optimal solution. Thus, expression  $Q/L_k(t)$  defines quantity of a pheromone with which the  $k$ -th ant marked an edge  $(i, j)$ .

### 4.3.1. The use of ant algorithm for the Traveling Salesman Problem

To check capability of ant algorithms to solve serious problems, we often use the known task of the traveling salesman problem – there is a set of the cities which the salesman should visit in such a way he arrives at each of them once and has to return to the city where he set off. A task is an optimization of a way so the passable distance can be the smallest one. It is one of the simplest problem definitions and we will try to solve it.

It is inconvenient to show the tasks connected with columns in the console therefore for this task we will choose graphical representation using Matplotlib, library for Python plotting. The reader can independently change and run an example, having taken the sources from appendix to this course. So, let's consider the realization of it in details.

First of all, we need to make some preparations. Let's see to the listing:

```
from math import sqrt
from random import random
import matplotlib.lines as lines
import matplotlib.pyplot as plt
import numpy as np
from tqdm import tqdm

MAX_CITIES = 10
MAX_DISTANCE = 1
MAX_TIME = 500 * MAX_CITIES
INIT_PHEROMONE = 1.0 / MAX_CITIES

MAX_ANTS = MAX_CITIES * MAX_CITIES
ALPHA = 1
BETA = 5
RHO = .5
QVAL = 100

ANTS = []
CITIES = []
DISTANCE = []
PHEROMONE = []
BEST = MAX_CITIES * MAX_DISTANCE
BEST_ANT = None
```

The first thing you need to do is import the necessary libraries, such as *NumPy* for fast computations, *Matplotlib* for graphical representation, and *TQDM* for showing beautiful progress, when our ants are walking. Keep in mind, that *TQDM* isn't included in the standard packages in Anaconda, and you must install it, for example by using the command line, *conda install -c conda-forge tqdm*.

First of all, we have two classes – *City* and *Ant* – the wrappers which are not bearing any special functionality.

```
class City(object):

    def __init__(self, x, y):
        self.x = x
        self.y = y

class Ant(object):

    def __init__(self, start_city):
        self.cur_city = start_city
        self.path = [start_city]
        self.tour_length = 0.

    def move_to_city(self, city):
        global DISTANCE, MAX_CITIES
        self.path.append(city)
        self.tour_length += DISTANCE[self.cur_city][city]
        if len(self.path) == MAX_CITIES:
            self.tour_length += DISTANCE[self.path[-1]][self.path[0]]
            self.cur_city = city

    def can_move(self):
        global MAX_CITIES
        return len(self.path) < MAX_CITIES

    def reset(self, city):
        self.cur_city = city
        self.path = [city]
        self.tour_length = 0.
```

The class *City* stores coordinates referring to “cities”, and the class *Ant* is used as an imitation of a real ant. The ant only can move, reset his state, and, of course, tell us, whether he can move or not.

Taking a step further we can find initialization code in function *init*, that is generating some random cities and standard pheromone value, assigned to each city. So, in final, now we have starting filed, where the ants will look for the best way.

```
def init():
    global DISTANCE, PHEROMONE, CITIES, ANTS

    for i in range(MAX_CITIES):
        DISTANCE.append([0.] * MAX_CITIES)
        PHEROMONE.append([INIT_PHEROMONE] * MAX_CITIES)

    for i in range(MAX_CITIES):
        CITIES.append(City(random() * MAX_DISTANCE, random() * MAX_DISTANCE))

    for i in range(MAX_CITIES):
        for j in range(MAX_CITIES):
            if i != j and DISTANCE[i][j] == 0.:
                xd = CITIES[i].x - CITIES[j].x
                yd = CITIES[i].y - CITIES[j].y
                distance = sqrt(xd * xd + yd * yd)
                DISTANCE[i][j], DISTANCE[j][i] = distance, distance

    to = 0
    for i in range(MAX_ANTS):
        ANTS.append(Ant(to))
        to += 1
    to = to % MAX_CITIES;
```

Furthermore, tables of distance between the cities and the table of pheromone trails are initialized. We initialize the table of pheromone trails by values, other than zero, so that the initial probability of the choice of given path can be positive.

The basic implementation of the traveling salesman algorithm is presented in the following listing. You can freely use this code in your projects, modify it at your discretion.

```
def ant_product(from_city, to_city, ph=None):
    global DISTANCE, PHEROMONE, ALPHA, BETA
    ph = ph or PHEROMONE[from_city][to_city]
    return (ph ** ALPHA) * \
        ((1. / DISTANCE[from_city][to_city]) ** BETA)
```



```

def select_next_city(ant):
    global MAX_CITIES, PHEROMONE, DISTANCE
    denom = 0.
    not_visited = []

    for to in range(MAX_CITIES):
        if to not in ant.path:
            ap = ant_product(ant.cur_city, to)
            not_visited.append((to, ap))
            denom += ap

    assert not_visited
    not_visited = [(val, ap / denom) for (val, ap) in not_visited]
    to = get_random(not_visited)
    return to
    i = 0
    while True:
        to, ap = not_visited[i]
        p = ap / denom
        if random() < p:
            break
        i += 1
        i = i % len(not_visited)

    if False and len(not_visited) == MAX_CITIES - 1:
        for to_city, ap in not_visited:
            print('%i %.03f %.01f %.02f % ('
                to_city, PHEROMONE[ant.cur_city][to_city], DISTANCE[ant.cur_city][to_city],
                ap /denom))
        print(to)
        raw_input()
    assert ant.cur_city != to
    return to

def simulate_ants():
    global ANTS, MAX_CITIES
    moving = 0

```

```

for ant in ANTS:
    if ant.can_move():
        ant.move_to_city(select_next_city(ant))
        moving += 1

return moving

def update_trails():
    global MAX_CITIES, PHEROMONE, RHO, INIT_PHEROMONE, ANTS

    # add new pheromone
    for ant in ANTS:
        pheromove_amount = QVAL / ant.tour_length

        for i in range(MAX_CITIES):
            if i == MAX_CITIES - 1:
                from_city = ant.path[i]
                to_city = ant.path[0]
            else:
                from_city = ant.path[i]
                to_city = ant.path[i + 1]
            assert from_city != to_city
            PHEROMONE[from_city][to_city] = PHEROMONE[from_city][to_city] * (1 - RHO) +
            pheromove_amount
            PHEROMONE[to_city][from_city] = PHEROMONE[from_city][to_city]

def restart_ants():
    global ANTS, BEST, BEST_ANT, MAX_CITIES
    to = 0

    for ant in ANTS:
        if ant.tour_length < BEST:
            BEST = ant.tour_length
            BEST_ANT = ant

    ant.reset(to)
    to += 1
    to = to % MAX_CITIES

```

Figure 4.10 shows the code of the main program written in the Jupiter notebook. The red frame highlights the code of the main loop, where the main calculations take place. During the calculations, a progress bar will be displayed.

```

if __name__ == '__main__':
    init()
    cur_time = 0
    for i in tqdm(range(MAX_TIME)):
        cur_time += 1
        if simulate_ants() == 0:
            update_trails()
            cur_time != MAX_TIME and restart_ants()

    x, y = [], []
    for i in BEST_ANT.path:
        city = CITIES[i]
        x.append(city.x / float(MAX_DISTANCE))
        y.append(city.y / float(MAX_DISTANCE))

    x.append(x[0])
    y.append(y[0])

    fig = plt.figure(figsize=(10, 10))

    ax = fig.add_subplot(1, 1, 1)

    x = np.array(x[:-1])
    y = np.array(y[:-1])
    line = lines.Line2D(x, y, mfc='red', ms=12, marker='o')
    ax.add_line(line)

    x = np.array([x[0], x[-1]])
    y = np.array([y[0], y[-1]])
    line = lines.Line2D(x, y, markevery=2, mfc='green', ms=14, marker='x')
    ax.add_line(line)

    plt.show()

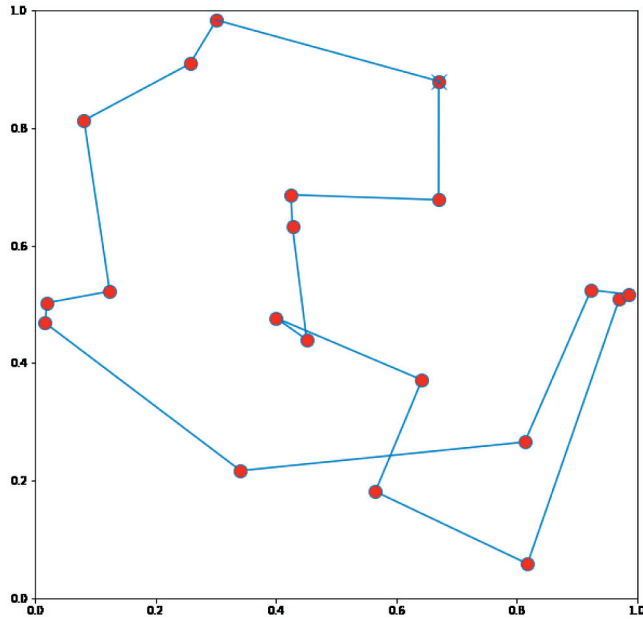
```

100% | ██████████ | 5000/5000 [00:03<00:00, 1331.60it/s]

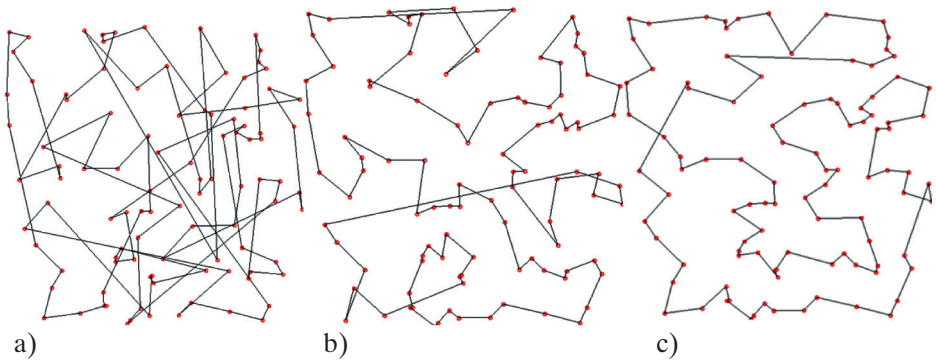
Fig. 4.10. Screenshot for solving a task of the traveler salesman problem in Jupiter Notebook

Finally, we get the final result in a graphical representation, as shown in the figure, we get the final result in a graphical representation, as shown in the Figure 4.11.

So, we considered a code which implements an algorithm of an ant algorithm. Let's see how it looks graphically. We start the application. We establish the number of the cities equal 100, and parameter beta equal 2. Further screenshots: of the beginning, the middle and the established process of the algorithm work are shown on Figure 4.12. As at each application run we start from the random city, readers will have pictures, different from provided in the book.



**Fig. 4.11.** Screenshot for solving a task of the traveler salesman problem



**Fig. 4.12.** Screenshots for solving a task of the traveler salesman problem:  
 a) the beging process; b) the middle process; c) the obtained result

It is visible that in the established decision there are still loops but not as repetitive as at the beginning.

It is possible to move on to changing settings of an algorithm. For example, if at this number of the cities we choose parameter beta equal 10, then the solution will be found quicker. So the search will get quicker. Also, it is possible to try to remove small loops, adding elite ants. It is possible to compare this results to the genetic algorithm. Try! Observe artificial life which is always entertaining – you can slightly feel like God...

## 5. Clustering methods

With the increase of information contents, the intelligence systems and processes are processed, stored and received as a result of work, during enterprises activity or research activity. This processing and the analysis become difficult. Thus, there is the need of initial information processing for its structuring, allocating of characteristic signs, generalizing, and sorting.

For this purpose applying the processes of classification and the clustering (see in Berry and Browne 2006, Han et al. 2011, Hastie et. al. 2009, Larose 2005, Roiger and Geatz 2003, Zaki and Meira 2014, Aggarwal and Reddy 2013, Barbakh et al. 2009, Diday and Simon 1976, Everit et al. 2001, Rokach 2009, Sneath and Sokal 1973, Gordon 1999, Idris 2014, Pedregosa 2011, Sammut and Webb 2017) allows to carry out preprocessing of information to its subsequent analysis (see more in Hand et al. 2011, Amorim and Mirkin 2013, Barbakh et al. 2009, Fraley and Raftery 1998, Grabmeier and Rudolph 2002, Hartigan 1975, Jain et al. 1999, Kaufmann and Rousseeuw 1990, Milligan and Cooper 1988 Pestunov et al. 2011, Sneath and Sokal 1973).

In practice all discriminant and clustering analysis (Hand et. al. 2011, Hartigan 1975, Jain et al. 199, Jain 2010, Gordon 1999, Pedregosa et al. 2011) are based on the compactness hypothesis. It can be formulated as follows: the objects belonging to one class have to be located compactly at least in one of possible spaces of the object description.

### **Classification**

A brief list of basic concepts.

Classification is a process of streamlining or distribution of objects (observations) in classes for the purpose of reflecting relations between them. The class is the set of the objects having the certain general sign distinguishing this set from other objects. According to classification division it is possible to accept the different signs depending on the classification purpose. The class always take the most important sign of the object answering to the classification purpose as a principle.

Classifying the object — means to specifying the class which this object belongs to. The name of the class is a result of object classification i.e. defined by the classification algorithm as a result of its application to this specific object.

Training of the qualifier — algorithm educating process, in case of the finite set of objects, for which it is known, to what classes they belong to. This set is called ‘training set’. Belonging to the class of other objects is not known.

### **Clustering**

Clustering is a process of splitting the set selection of objects (observations) into the subsets (as a rule, not crossed) called clusters so that each cluster consists of similar objects, and objects of different clusters are differ significantly.

One of the purposes of the clustering is identification of internal communications between given, by a way of definition, cluster structure. Splitting observations into groups of similar objects allows to simplify further data processing and decision making, applying the method of the analysis – “divide et impera” to each cluster (i.e. strategy “divide and conquer”).

The one of more known applications of the clustering involves solving the problem of data compression. If an initial selection is too big, then it is possible to reduce it, leaving some of the most characteristic representatives from each cluster.

The clustering is other sphere of using detection of novelty in the studied objects set. Typical objects which do not manage to be attached to one of clusters are allocated. For solving problems by methods of cluster analysis, it is necessary to fix the quantity of clusters in advance. In one case, number of clusters can be reduced. In another case, it is more important to provide high degree of objects similarity in each cluster, and number of clusters can be as big as necessary. In the third case, the separate objects, which do not fit in any of clusters are the most important (see for instance in Jain 2010, Grabmeier and Rudolph 2002, Mirkin 2005, Pestunov et al. 2011, Rokach 2009).

## **5.1. Clustering. General concepts**

Let the set of objects  $\mathfrak{S} = \{x_i\}_{i=1}^n$  is presented by the set of attributes  $x_i = \{t_1^i, t_2^i, \dots, t_m^i\}$ , where  $t_v^i$  accepts values from the set  $T_v^i$ . The problem of the clustering consists of creating the sets  $C = \{c_v\}_{v=1}^k$  and displaying  $F: \mathfrak{S} \rightarrow C$  as the set of objects on the set of clusters.

The cluster contains objects from  $\mathfrak{S}$  similar (by the set criterion) to each other

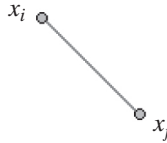
$$x_i \in c_v, x_j \in c_v \Rightarrow d(x_i, x_j) < \varepsilon,$$

where  $d(\cdot, \cdot)$  is a proximity measure between objects (distance), and  $\varepsilon$  – the maximum value the threshold creating one cluster.

The most used measures of proximity is the concept of distance between two points  $\|x_i - x_j\|$ . Let's give some most commonly used definitions of distance.

- Euclidean (mean square or  $\ell_2$ ) distance (see Fig. 5.1):

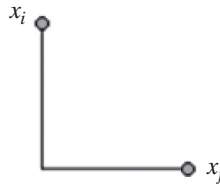
$$\|x_i - x_j\|_2 = \sqrt{\sum_{v=0}^m (t_v^i - t_v^j)^2}$$



**Fig. 5.1.** An illustration of Euclidean distance

- distance  $\ell_1$  (in English-speaking literature – ‘manhattan’ or city distance) (see Fig. 5.2):

$$\|x_i - x_j\|_1 = \sum_{v=0}^m |t_v^i - t_v^j|$$

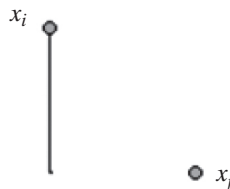


**Fig. 5.2.** An illustration of distance  $\ell_1$

it is used in the case when it is necessary to reduce influence of separate emissions;

- chebyshev (uniform or  $\ell_\infty$ ) distance (see Fig. 5.3):

$$\|x_i - x_j\|_\infty = \max_{v=0, \dots, m} |t_v^i - t_v^j|$$



**Fig. 5.3.** An illustration of Chebyshev distance

it is used if it is necessary to increase the influence of separate emissions;

- Mahalanobis distance

$$\|x_i - x_j\|_M = \sqrt{(x_i - x_j)\Sigma^{-1}(x_i - x_j)^T},$$

it is used when it is necessary to consider a correlation for a specific class.  $\Sigma$  is the complete correlation matrix.

Let's point out that in many cases instead of distance as criterion of proximity the value of the cosine of the angle between two vectors is used

$$\cos \phi_{i,j} = \frac{x_i^T x_j}{\|x_i\|_2 \|x_j\|_2}.$$

A correlation factor is also often used

$$\sigma_{i,j} = \frac{\sum_{v=1}^n (t_i^v - \mu_i)(t_j^v - \mu_j)}{\sqrt{\sum_{v=1}^n (t_i^v - \mu_i)^2 \sum_{v=1}^n (t_j^v - \mu_j)^2}}.$$

It is very important for the problem relating to clustering that all data were commensurable i.e. that it did not turn out that one part of data is measured in packing units and the other in kilograms, in other words, at the first stage of the clustering it is always necessary to carry out data normalization. Besides, when holding the procedure based on clustering it is necessary to determine either number of clusters, or a criterion, characterizing proximity of elements in the cluster or remoteness of clusters from each other in advance.(see more in Fraley and Raftery 1998, Kaufman and Rosseeuw 1990, Milligan and Cooper 1985, Mirkin 2005, Nguyen and Doan 2012, Sneath and Sokal 1973).

Clustering methods can be divided into two classes – hierarchical and nonhierarchical. In non-hierarchical algorithms we use the stop condition (in the sense of epsilon in the iteration difference) and a fixed number of clusters. The basis of these algorithms is the hypothesis of rather small number of the hidden factors which define structure of communication between signs. Hierarchical algorithms are not tied on quantity of clusters. Its characteristic is determined by dynamics of merge and division of clusters during creation of the tree of the enclosed clusters (dendrogram). In turn, hierarchical algorithms are divided into agglomerative methods, which are built by combining elements, i.e. reducing the number of clusters, and divisive method, based



on division of existing groups (clusters) (see, for example, Everit et al. 2001, Hartigan 1975, Lance and Williams 1967, 1968a, 1968b, Macnaughton-Smith et al. 1964 Mirkin 2005, Roux 2015, *Diana algorithm*).

## 5.2. Hierarchical methods

### 5.2.1. Hierarchical methods. Agglomerative algorithms

On the first step each element  $\mathfrak{S}$  is a separate cluster  $c_i^0 = \{x_i\}$ . On each subsequent step two closest clusters  $c_i^v$  and  $c_i^\mu$  combine in one, so the set formed of  $n - 1$  clusters contains  $c_1^1 = \{c_v^0, c_\mu^0\}$ ,  $c_i^1 = \{c_1^0 (i \neq \{v, \mu\})\}$  (see Fig. 5.4).

Further this process repeats. In the end we will receive one cluster matching with all elements of  $\mathfrak{S}$ .

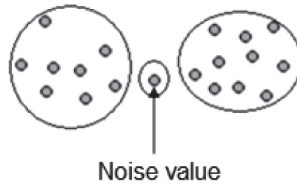


Fig. 5.4. Input data

At the merging  $v$  and  $\mu$  clusters in  $i$ -th step it is necessary to calculate distance from the new cluster to other clusters. Traditionally, for recalculation of distance during merge of clusters old values of distances are used (see Hartigan 1975, Kaufman and Rousseeuw 1990, Lance and Williams 1968a, 1968b, Mirkin 2005, Pedregosa et al. 2011). As a rule, at the same time use the following criteria (p. 1–4).

- 1) Minimum distance (see Fig. 5.5):

$$d_{\min}(c_i, c_j) = \min\{\|x - y\| \mid x \in c_i, y \in c_j\}.$$

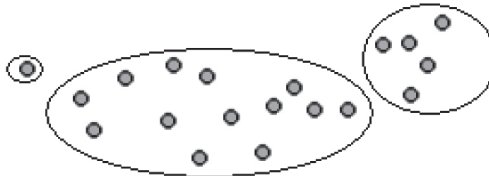
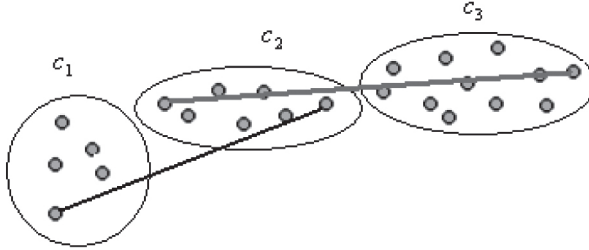


Fig. 5.5. Result of application of the agglomerative clustering with the minimum distance

This algorithm promotes growth of the extended clusters. To algorithm shortcomings, first of all, it is necessary to refer sensitivity to noise.

2) Maximum distance (see Fig. 5.6):

$$d_{\max}(c_i, c_j) = \max\{x - y \mid x \in c_i, y \in c_j\}.$$



**Fig. 5.6.** Result of application of the agglomerative clustering with the maximum distance

The algorithm on the basis of the maximum distance promotes formation of compact clusters. The effect of destruction of the extended clusters belongs to shortcomings.

3) Average distance

$$\tilde{d}(c_i, c_j) = \frac{1}{n_i n_j} \sum_{x \in c_i} \sum_{y \in c_j} \|x - y\|.$$

4) Distance between centers clusters

$$d_{\mu}(c_i, c_j) = \|\mu_i - \mu_j\|.$$

The two algorithms mentioned above are based on average distance. From the computing standpoint the stability and efficiency are appropriate in practice. The algorithms based on distance between centers of clusters are more effective.

## 5.2.2. Hierarchical methods. Divisive algorithms

The ideology of divisive algorithms is dual to agglomerative ones. In the first step, all elements belong to one set  $\mathfrak{S}$ . In next steps, one of the existing clusters is divided into two separate subsets.

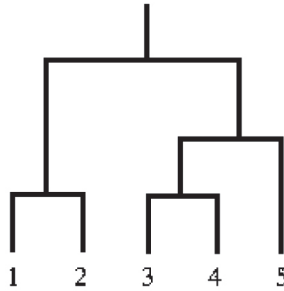


Fig. 5.7. An illustration of the divisive algorithm of the clustering

The simplest algorithm, among such algorithms, was described a long time ago by Macnaughton-Smith et al. in 1964 (Macnaughton-Smith et al. 1964) and reused by Kaufman and Rosseeuw 1990, Lance and Williams 1968b, Roux 2015. The point is that in the beginning we choose an element from cluster  $c_0^1 = \mathcal{J}$ , which is the furthest from the center of the cluster, and this element creates the new cluster  $c_2^1$ , whereas the remained elements, create  $c_1^1 = c_0^1 \setminus c_2^1$ . On each subsequent step the element from  $c_1^1$ , for which the difference between distance to the center from the cluster  $c_2^1$  and the cluster  $c_1^1$  is the greatest, is attached to  $c_2^1$ . This process is being continued until the difference becomes negative, i.e. until all elements are chosen from  $c_1^1$ . Splitting one cluster into two will be the result. Further, this iterative process continues. The choice of the split cluster can be carried out proceeding from different reasons, for example, the cluster with the largest diameter is chosen i.e.  $\max_{i,j} \|x_i - x_j\|$ , where elements  $x_i$ , and  $x_j$  lie in one cluster. As a rule, divisive algorithms are used in case of small size selections. An example is shown in Figure 5.7.

### 5.3. Examples in Python – clustering hierarchical methods

Several examples are given below. The following examples were inspired by Peregosa et al. 2011.

```
#####
# Example 5.1
# Ward's clustering with different distances
# Dataset from table 2.2
#####

import numpy as np
import pandas as pd
```

```

import scipy as sc
import matplotlib.pyplot as plt
from sklearn.metrics.pairwise import pairwise_distances
from sklearn.cluster import AgglomerativeClustering
from scipy.spatial.distance import pdist, squareform

# read data
x = [1,2,3,4,5,6,7,8]
y = [2,3,2,4,4,7,6,7]
# data from table 2.2
x1 = (x - np.mean(x))/np.std(x)
y1 = (y - np.mean(y))/np.std(y)

dane = pd.DataFrame({
    'x1': np.array(x1),
    'y1': np.array(y1),
})

# distances
S1 = pairwise_distances(dane, metric="euclidean")
S2 = pairwise_distances(dane, metric="manhattan")
S3 = pairwise_distances(dane, metric="cosine")
S4 = squareform(pdist(dane,'chebyshev'))

def plot_clusters(ax, title, fontsize=12):
    if title is not None:
        plt.title(title, size=fontsize)
    plt.tight_layout()

plt.close('all')
fig = plt.figure()

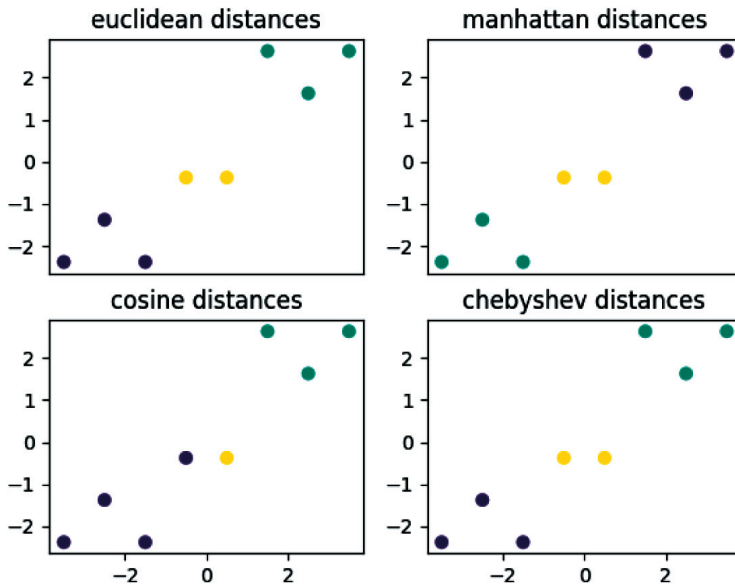
# clustering and plot clusters
plt.subplot(221)
clustering = AgglomerativeClustering(connectivity=S1,linkage='ward', n_clusters=3)
y_pred=clustering.fit_predict(S1)
plt.scatter(dane.x1,dane.y1, c=y_pred)
plt.title("euclidean distances")
plt.xticks([])

```

```
plt.subplot(222)
clustering = AgglomerativeClustering(connectivity=S2,linkage='ward', n_clusters=3)
y_pred=clustering.fit_predict(S2)
plt.scatter(dane.x1,dane.y1, c=y_pred)
plt.title("manhattan distances")
plt.xticks([])
```

```
plt.subplot(223)
clustering = AgglomerativeClustering(connectivity=S3,linkage='ward', n_clusters=3)
y_pred=clustering.fit_predict(S3)
plt.scatter(dane.x1,dane.y1, c=y_pred)
plt.title("cosine distances")
plt.xticks([])
```

```
plt.subplot(224)
clustering = AgglomerativeClustering(connectivity=S4,linkage='ward', n_clusters=3)
y_pred=clustering.fit_predict(S4)
plt.scatter(dane.x1,dane.y1, c=y_pred)
plt.title("chebyshev distances")
plt.xticks([])
plt.show()
```



```

#####
# Example 5.2.1
# Hierarchical clustering with the different distances – Ward's method
# Masterpieces dataset (from B.Mirkin Clustering for data mining)
#####
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
from sklearn.metrics.pairwise import pairwise_distances
from sklearn.cluster import AgglomerativeClustering
from scipy.spatial.distance import pdist, squareform

# Masterpieces dataset
x = [19,29.4,23.9,18.4,25.7,12.1,23.9,27.2]
y = [43.7,36,38,27.9,22.3,16.9,30.2,58]
plt.scatter(x,y)
# data from table 2.2
x1 = (x - np.mean(x))/np.std(x)
y1 = (y - np.mean(y))/np.std(y)

dane = pd.DataFrame({
    'x1': np.array(x1),
    'y1': np.array(y1),
})

S1 = pairwise_distances(dane, metric="euclidean")
S2 = pairwise_distances(dane, metric="manhattan")
S3 = pairwise_distances(dane, metric="cosine")
S4 = squareform(pdist(dane,'chebyshev'))

plt.close('all')
fig = plt.figure()

# clustering and plot clusters
plt.subplot(221)
clustering = AgglomerativeClustering(connectivity=S1,linkage='ward', n_clusters=3)
y_pred=clustering.fit_predict(S1)
plt.scatter(dane.x1,dane.y1, c=y_pred)
plt.title("euclidean distances")
plt.xticks([])

```

```

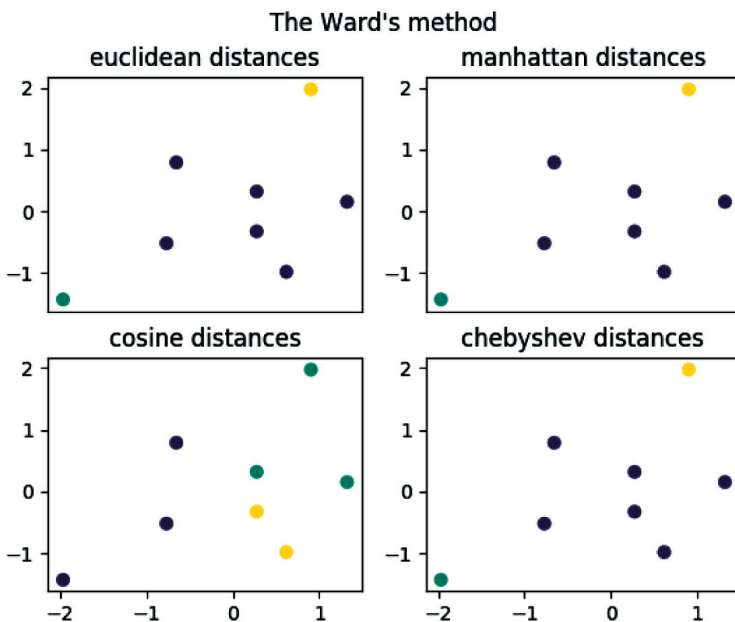
plt.subplot(222)
clustering = AgglomerativeClustering(connectivity=S2,linkage='ward', n_clusters=3)
y_pred=clustering.fit_predict(S2)
plt.scatter(dane.x1,dane.y1, c=y_pred)
plt.title("manhattan distances")
plt.xticks([])

plt.subplot(223)
clustering = AgglomerativeClustering(connectivity=S3,linkage='ward', n_clusters=3)
y_pred=clustering.fit_predict(S3)
plt.scatter(dane.x1,dane.y1, c=y_pred)
plt.title("cosine distances")

plt.subplot(224)
clustering = AgglomerativeClustering(connectivity=S4,linkage='ward', n_clusters=3)
y_pred=clustering.fit_predict(S4)
plt.scatter(dane.x1,dane.y1, c=y_pred)
plt.title("chebyshev distances")

fig.suptitle('The Ward\'s method')
plt.show()

```



```

#####
# Example 5.2.2
# Hierarchical clustering with the different distances – average method
# Masterpieces dataset (from B.Mirkin Clustering for data mining)
#####
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
from sklearn.metrics.pairwise import pairwise_distances
from sklearn.cluster import AgglomerativeClustering
from scipy.spatial.distance import pdist, squareform

# Masterpieces dataset
x = [19,29.4,23.9,18.4,25.7,12.1,23.9,27.2]
y = [43.7,36,38,27.9,22.3,16.9,30.2,58]
plt.scatter(x,y)
# data from table 2.2
x1 = (x - np.mean(x))/np.std(x)
y1 = (y - np.mean(y))/np.std(y)

dane = pd.DataFrame({
    'x1': np.array(x1),
    'y1': np.array(y1),
})

S1 = pairwise_distances(dane, metric="euclidean")
S2 = pairwise_distances(dane, metric="manhattan")
S3 = pairwise_distances(dane, metric="cosine")
S4 = squareform(pdist(dane,'chebyshev'))

plt.close('all')
fig = plt.figure()

# clustering and plot clusters
plt.subplot(221)
clustering = AgglomerativeClustering(connectivity=S1,linkage='average', n_clusters=3)
y_pred=clustering.fit_predict(S1)
plt.scatter(dane.x1,dane.y1, c=y_pred)
plt.title("euclidean distances")
plt.xticks([])

```



```

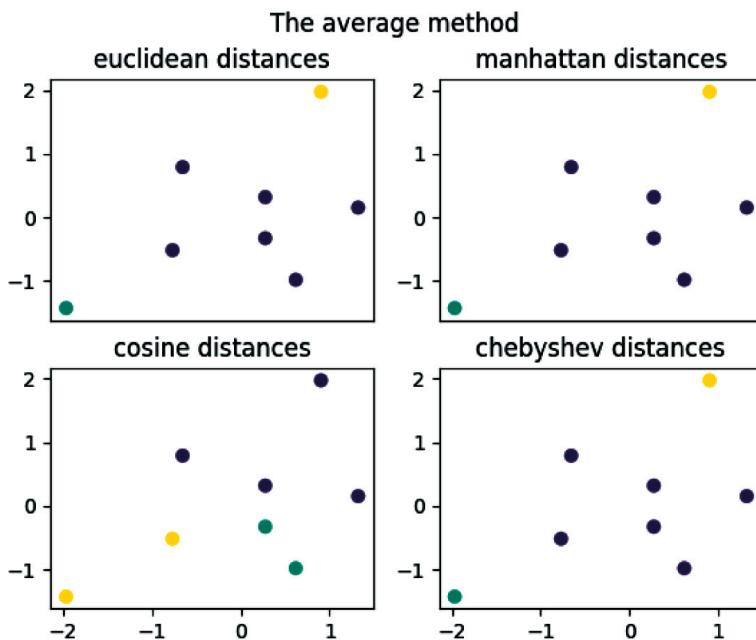
plt.subplot(222)
clustering = AgglomerativeClustering(connectivity=S2,linkage='average', n_clusters=3)
y_pred=clustering.fit_predict(S2)
plt.scatter(dane.x1,dane.y1, c=y_pred)
plt.title("manhattan distances")
plt.xticks([])

plt.subplot(223)
clustering = AgglomerativeClustering(connectivity=S3,linkage='average', n_clusters=3)
y_pred=clustering.fit_predict(S3)
plt.scatter(dane.x1,dane.y1, c=y_pred)
plt.title("cosine distances")

plt.subplot(224)
clustering = AgglomerativeClustering(connectivity=S4,linkage='average', n_clusters=3)
y_pred=clustering.fit_predict(S4)
plt.scatter(dane.x1,dane.y1, c=y_pred)
plt.title("chebyshev distances")

fig.suptitle("The average method")
plt.show()

```



```

#####
# Example 5.2.3
# Hierarchical clustering with the different distances – complete method
# Masterpieces dataset (from B.Mirkin Clustering for data mining)
#####
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
from sklearn.metrics.pairwise import pairwise_distances
from sklearn.cluster import AgglomerativeClustering
from scipy.spatial.distance import pdist, squareform

# Masterpieces dataset
x = [19,29.4,23.9,18.4,25.7,12.1,23.9,27.2]
y = [43.7,36,38,27.9,22.3,16.9,30.2,58]
plt.scatter(x,y)
# data from table 2.2
x1 = (x - np.mean(x))/np.std(x)
y1 = (y - np.mean(y))/np.std(y)

dane = pd.DataFrame({
    'x1': np.array(x1),
    'y1': np.array(y1),
})

S1 = pairwise_distances(dane, metric="euclidean")
S2 = pairwise_distances(dane, metric="manhattan")
S3 = pairwise_distances(dane, metric="cosine")
S4 = squareform(pdist(dane,'chebyshev'))

plt.close('all')
fig = plt.figure()

# clustering and plot clusters
plt.subplot(221)
clustering = AgglomerativeClustering(connectivity=S1,linkage='complete', n_clusters=3)
y_pred=clustering.fit_predict(S1)
plt.scatter(dane.x1,dane.y1, c=y_pred)
plt.title("euclidean distances")
plt.xticks([])

```

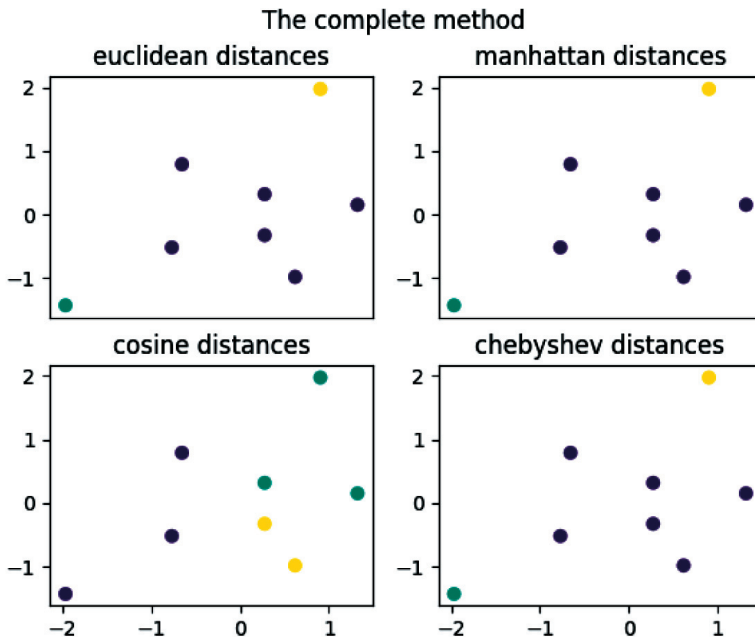
```
plt.subplot(222)
clustering = AgglomerativeClustering(connectivity=S2,linkage='complete', n_clusters=3)
y_pred=clustering.fit_predict(S2)
plt.scatter(dane.x1,dane.y1, c=y_pred)
plt.title("manhattan distances")
plt.xticks([])
```

```
plt.subplot(223)
clustering = AgglomerativeClustering(connectivity=S3,linkage='complete', n_clusters=3)
y_pred=clustering.fit_predict(S3)
plt.scatter(dane.x1,dane.y1, c=y_pred)
plt.title("cosine distances")
```

```
plt.subplot(224)
clustering = AgglomerativeClustering(connectivity=S4,linkage='complete', n_clusters=3)
y_pred=clustering.fit_predict(S4)
plt.scatter(dane.x1,dane.y1, c=y_pred)
plt.title("chebyshev distances")
```

```
fig.suptitle("The complete method")
```

```
plt.show()
```



```

#####
# Example 5.3.1
# Hierarchical clustering
# Generating dataset (blobs)
#####
import matplotlib.pyplot as plt
import numpy as np
from sklearn.cluster import AgglomerativeClustering
from sklearn import datasets
from sklearn.preprocessing import StandardScaler

n_samples = 2500
n_features = 2
centers = [(-5, -5), (-4, -2), (5, 5)]

blobs = datasets.make_blobs(n_samples=n_samples,n_features=n_features,cluster_std=1.0,
                           centers=centers, shuffle=False, random_state= 120)

transformation = [[ 0.60834549, -0.63667341], [-0.40887718, 0.85253229]]
X, y = blobs
X_flattened_blobs = np.dot(X, transformation)

# possible linkage is 'ward', 'average', 'complete'
linkage = 'ward'

plt.close('all')
fig = plt.figure()
cmap = plt.cm.get_cmap("Dark2")

plt.subplot(221)
plt.scatter(X[:, 0], X[:, 1], color=cmap(.5/3.))
plt.title("dataset blobs")
plt.subplot(222)
plt.scatter(X_flattened_blobs[:, 0], X_flattened_blobs[:, 1], color=cmap(.75/3.))
plt.title("dataset transforming blobs")

clustering = AgglomerativeClustering(linkage=linkage, n_clusters=3)
X, y = blobs
X_stand = StandardScaler().fit_transform(X)

```

```

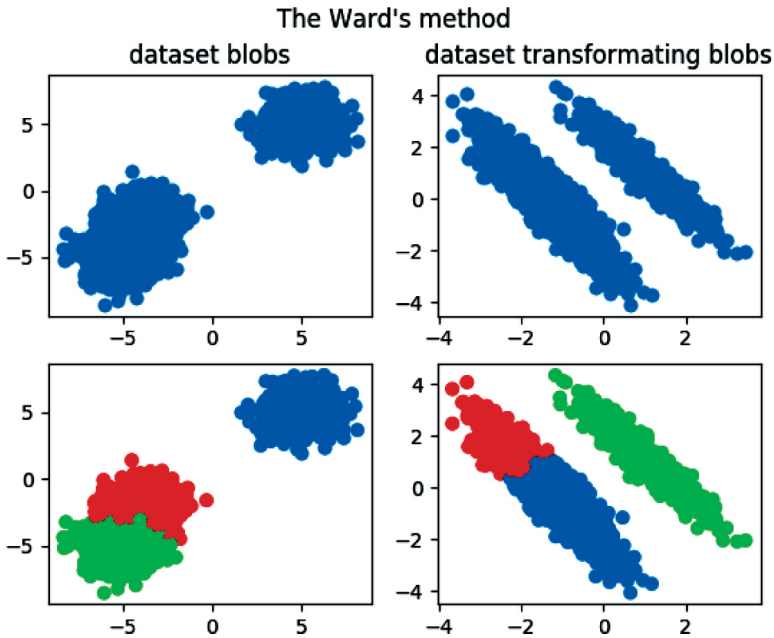
clustering.fit(X_stand)
y_pred = clustering.fit_predict(X_stand)
plt.subplot(223)
plt.scatter(X[:, 0], X[:, 1], color=cmap((y_pred+.3)/3.))

X_stand = StandardScaler().fit_transform(X_flattened_blobs)
clustering.fit(X)
y_pred = clustering.fit_predict(X_stand)
plt.subplot(224)
plt.scatter(X_flattened_blobs[:, 0], X_flattened_blobs[:, 1], color=cmap((y_pred+.2)/3.))

fig.suptitle("The Ward's method")

plt.show()

```



```

#####
# Example 5.3.2
# Hierarchical clustering
# Generating dataset (blobs with different variance)
#####
import numpy as np
import matplotlib.pyplot as plt

```

```

from sklearn import datasets
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import AgglomerativeClustering
np.random.seed(0)

n_samples = 2500
n_features = 2
centers = [(-5, -5), (-4, 2), (5, 5)]

blobs = datasets.make_blobs(n_samples=n_samples, n_features=n_features, cluster_std=
    [1.0, 0.5, 2.5],
    centers=centers, shuffle=False, random_state= 120)

transformation = [[ 0.60834549, -0.63667341], [-0.40887718, 0.85253229]]
X, y = blobs
X_flattened_blobs = np.dot(X, transformation)
# possible linkage is 'ward', 'average', 'complete'
linkage = 'ward'

plt.close('all')
fig = plt.figure()
cmap = plt.cm.get_cmap('Dark2')

plt.subplot(221)
plt.scatter(X[:, 0], X[:, 1], color= cmap((y_pred+.75)/3.))
plt.title("blobs")
plt.subplot(222)
plt.scatter(X_flattened_blobs[:, 0], X_flattened_blobs[:, 1], color= cmap((.85)/3.))
plt.title("transforming blobs")

clustering = AgglomerativeClustering(linkage=linkage, n_clusters=3)
X, y = blobs
X_stand = StandardScaler().fit_transform(X)
clustering.fit(X_stand)
y_pred = clustering.fit_predict(X_stand)
plt.subplot(223)
plt.scatter(X[:, 0], X[:, 1], color= cmap((y_pred+.85)/3.))

```

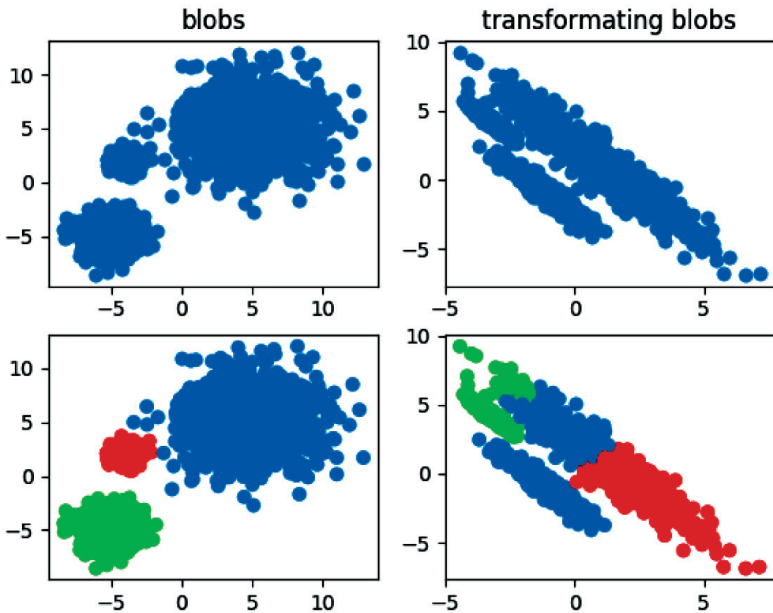
```

X_stand = StandardScaler().fit_transform(X_flattened_blobs)
clustering.fit(X)
y_pred = clustering.fit_predict(X_stand)
plt.subplot(224)
plt.scatter(X_flattened_blobs[:, 0], X_flattened_blobs[:, 1], color= cmap((y_pred+.85)/3.))

fig.suptitle("The Ward's method for blobs with different variance")

plt.show()

```



```

#####
# Example 5.3.3
# Hierarchical clustering
# Generating dataset (blobs with different variance)
#####
import numpy as np
import matplotlib.pyplot as plt

from sklearn import datasets
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import AgglomerativeClustering
np.random.seed(0)

```

```

n_samples = 2500
n_features = 2
centers = [(-5, -5), (-4, 2), (5, 5)]

blobs = datasets.make_blobs(n_samples=n_samples,n_features=n_features,cluster_std=
    [1.0, 0.5, 2.5],
    centers=centers, shuffle=False, random_state= 120)

transformation = [[ 0.60834549, -0.63667341], [-0.40887718, 0.85253229]]

X, y = blobs
X_filtered = np.vstack((X[y == 0][:500], X[y == 1][:100], X[y == 2][:10]))
X_flattened_blobs = np.dot(X_filtered, transformation)

# possible linkage is 'ward', 'average', 'complete'
linkage = 'ward'

plt.close('all')
fig = plt.figure()
cmap = plt.cm.get_cmap('hsv')

plt.subplot(221)
plt.scatter(X_filtered[:, 0], X_filtered[:, 1], color=cmap((.75)/3.))
plt.title("blobs")
plt.subplot(222)
plt.scatter(X_flattened_blobs[:, 0], X_flattened_blobs[:, 1], color=cmap((.75)/3.))
plt.title("transforming blobs")

clustering = AgglomerativeClustering(linkage=linkage, n_clusters=3)

X_stand = StandardScaler().fit_transform(X_filtered)
clustering.fit(X_stand)
y_pred = clustering.fit_predict(X_stand)
plt.subplot(223)
plt.scatter(X_filtered[:, 0], X_filtered[:, 1], color=cmap((y_pred+.75)/3.))

X_stand = StandardScaler().fit_transform(X_flattened_blobs)
clustering.fit(X_stand)

```



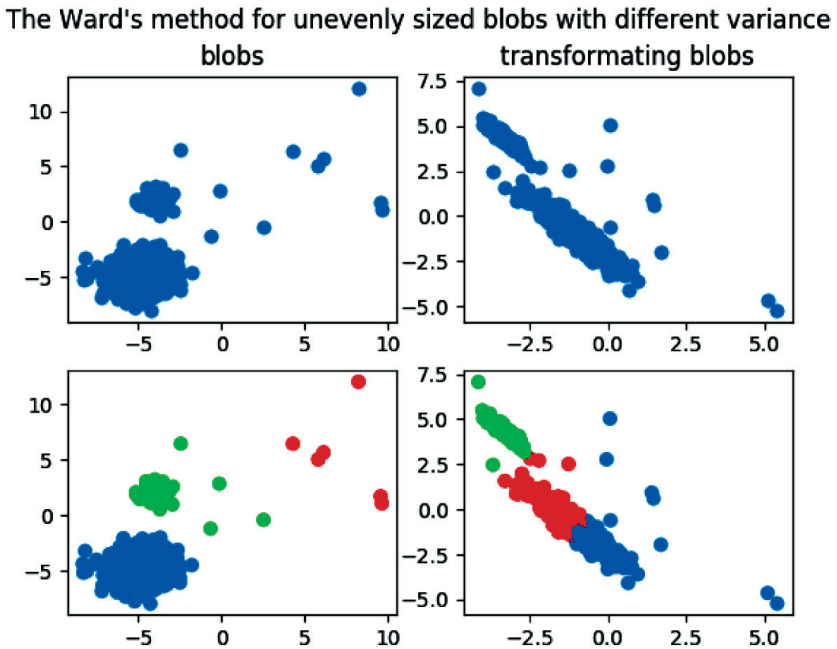
```

y_pred = clustering.fit_predict(X_stand)
plt.subplot(224)
plt.scatter(X_flattened_blobs[:, 0], X_flattened_blobs[:, 1], color=cmap((y_pred+.75)/3.))

fig.suptitle("The Ward's method for unevenly sized blobs with different variance")

plt.show()

```



```

#####
# Example 5.4
# Hierarchical clustering
# Generating dataset (no structure)
#####
import numpy as np
import matplotlib.pyplot as plt

from sklearn.preprocessing import StandardScaler
from sklearn.cluster import AgglomerativeClustering
np.random.seed(0)

```

```

n_samples = 2500
n_features = 2

no_structure = np.random.rand(n_samples, 2), None

# possible linkage is 'ward', 'average', 'complete'
plt.close('all')
fig = plt.figure()
cmap = plt.cm.get_cmap('hsv')

X, y = no_structure
X_stand = StandardScaler().fit_transform(X)
plt.subplot(221)
plt.scatter(X[:, 0], X[:, 1], color=cmap((.75)/3.))
plt.title("Dataset no structure")
plt.xticks([])

clustering = AgglomerativeClustering(linkage='ward', n_clusters=3)
clustering.fit(X_stand)
y_pred = clustering.fit_predict(X_stand)
plt.subplot(222)
plt.scatter(X[:, 0], X[:, 1], color=cmap((y_pred+.75)/3.))
plt.title("Ward's method")
plt.xticks([])

clustering = AgglomerativeClustering(linkage='average', n_clusters=3)
clustering.fit(X_stand)
y_pred = clustering.fit_predict(X_stand)
plt.subplot(223)
plt.scatter(X[:, 0], X[:, 1], color=cmap((y_pred+.75)/3.))
plt.title("Average method")

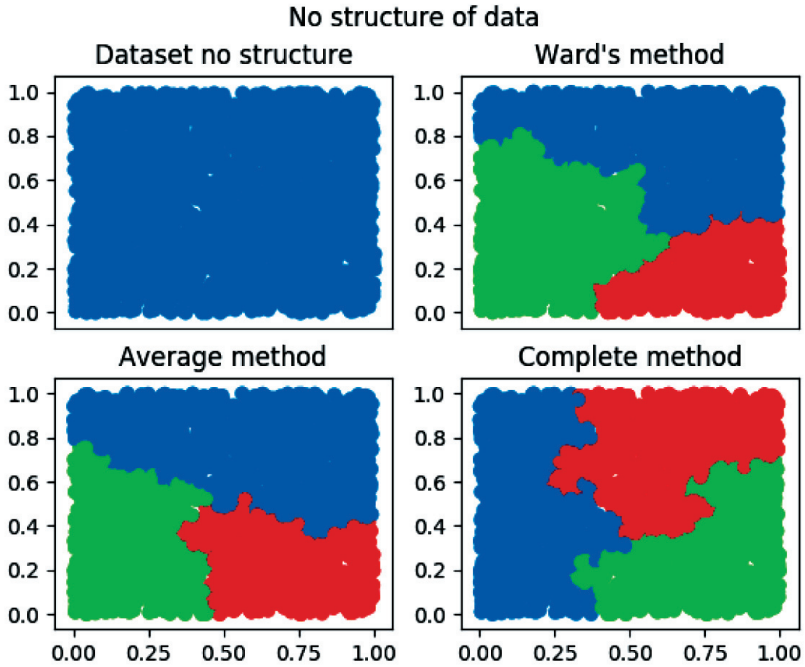
clustering = AgglomerativeClustering(linkage='complete', n_clusters=3)
clustering.fit(X_stand)
y_pred = clustering.fit_predict(X_stand)
plt.subplot(224)
plt.scatter(X[:, 0], X[:, 1], color=cmap((y_pred+.75)/3.))

```

```
plt.title("Complete method")
```

```
fig.suptitle("No structure of data")
```

```
plt.show()
```



```
#####  
# Example 5.5  
# Hierarchical clustering  
# Generating dataset (noisy circles)  
#####  
import numpy as np  
import matplotlib.pyplot as plt  
  
from sklearn.preprocessing import StandardScaler  
from sklearn import datasets  
from sklearn.cluster import AgglomerativeClustering  
  
np.random.seed(0)
```

```

n_samples = 2500
n_features = 2

noisy_circles = datasets.make_circles(n_samples=n_samples, factor=.5, noise=.05)

# possible linkage is 'ward', 'average', 'complete'
plt.close('all')
fig = plt.figure()
cmap = plt.cm.get_cmap('hsv')

X, y = noisy_circles
X_stand = StandardScaler().fit_transform(X)
plt.subplot(221)
plt.scatter(X[:, 0], X[:, 1], color=cmap(.75/3.))
plt.title("Dataset noisy circles")
plt.xticks([])

clustering = AgglomerativeClustering(linkage='ward', n_clusters=2)
clustering.fit(X_stand)
y_pred = clustering.fit_predict(X_stand)
plt.subplot(222)
plt.scatter(X[:, 0], X[:, 1], color=cmap((y_pred+.75)/3.))
plt.title("Ward's method")
plt.xticks([])

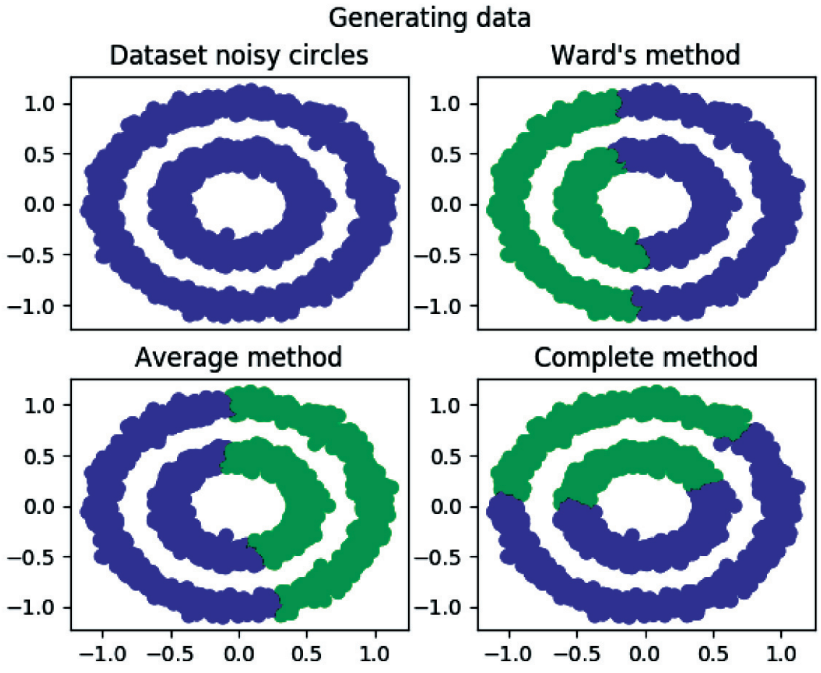
clustering = AgglomerativeClustering(linkage='average', n_clusters=2)
clustering.fit(X_stand)
y_pred = clustering.fit_predict(X_stand)
plt.subplot(223)
plt.scatter(X[:, 0], X[:, 1], color=cmap((y_pred+.75)/3.))
plt.title("Average method")

clustering = AgglomerativeClustering(linkage='complete', n_clusters=2)
clustering.fit(X_stand)
y_pred = clustering.fit_predict(X_stand)
plt.subplot(224)

```

```
plt.scatter(X[:, 0], X[:, 1], color=cmap((y_pred+.75)/3.))
plt.title("Complete method")
```

```
fig.suptitle("Generating data")
plt.show()
```



```
#####
# Example 5.6
# Hierarchical clustering
# Generating dataset (noisy moons)
#####
import numpy as np
import matplotlib.pyplot as plt

from sklearn.preprocessing import StandardScaler
from sklearn import datasets
from sklearn.cluster import AgglomerativeClustering

np.random.seed(0)
```

```

n_samples = 2500
n_features = 2

noisy_moons = datasets.make_moons(n_samples=n_samples, noise=.05)
plt.close('all')
fig = plt.figure()
cmap = plt.cm.get_cmap('hsv')

X, y = noisy_moons
X_stand = StandardScaler().fit_transform(X)
plt.subplot(221)
plt.scatter(X[:, 0], X[:, 1], color=cmap(.9/3.))
plt.title("Dataset noisy moons")
plt.xticks([])

clustering = AgglomerativeClustering(linkage='ward', n_clusters=2)
clustering.fit(X_stand)
y_pred = clustering.fit_predict(X_stand)
plt.subplot(222)
plt.scatter(X[:, 0], X[:, 1], color=cmap((y_pred+.9)/3.))
plt.title("Ward's method")
plt.xticks([])

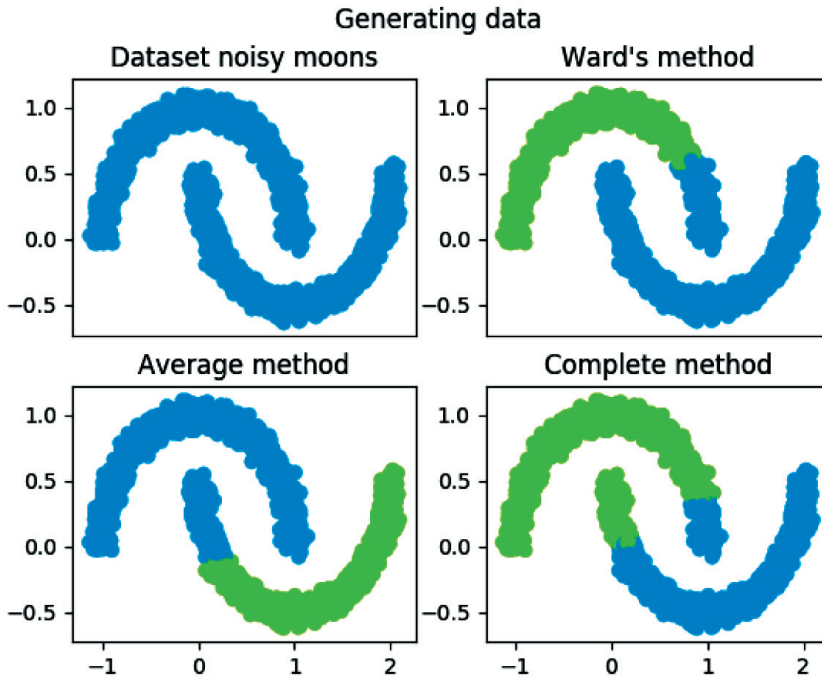
clustering = AgglomerativeClustering(linkage='average', n_clusters=2)
clustering.fit(X_stand)
y_pred = clustering.fit_predict(X_stand)
plt.subplot(223)
plt.scatter(X[:, 0], X[:, 1], color=cmap((y_pred+.9)/3.))
plt.title("Average method")

clustering = AgglomerativeClustering(linkage='complete', n_clusters=2)
clustering.fit(X_stand)
y_pred = clustering.fit_predict(X_stand)
plt.subplot(224)
plt.scatter(X[:, 0], X[:, 1], color=cmap((y_pred+.9)/3.))

```

```
plt.title("Complete method")

fig.suptitle("Generating data")
plt.show()
```



```
#####
# Example 5.7.1
# Hierarchical clustering
# Digits dataset, Ward's method, two ways of scaling
#####

import numpy as np
from matplotlib import pyplot as plt
from sklearn import manifold, datasets
from sklearn.cluster import AgglomerativeClustering
from sklearn.preprocessing import StandardScaler
#import tensorflow as tf

digits = datasets.load_digits(n_class=10)
```

```

X = digits.data[:1200]

y = digits.target
n_samples, n_features = X.shape
print('be patient the projection on a two-dimensional plane takes a few minutes...\n')
# projection on a two-dimensional plane it takes a few minutes
np.random.seed(0)
seed = np.random.RandomState(seed=3)
X_set = manifold.SpectralEmbedding(n_components=2).fit_transform(X)
X_stand = StandardScaler().fit_transform(X_set)
X_set2 = manifold.MDS(n_components=2,max_iter=3000, eps=1e-9,
                      random_state=seed).fit_transform(X)
print('It is done !')

s=25
plt.close('all')
fig = plt.figure()
cmap = plt.cm.get_cmap('hsv')

plt.subplot(221)
plt.scatter(X_stand[:,0], X_stand[:,1], color=(.2,.4,.7), s=s) #lw =0
plt.xticks([])
plt.subplot(222)
plt.scatter(X_set2[:,0], X_set2[:,1], color=(.85,.25,.05), s=s) #lw =0
plt.xticks([])

# clustering
clustering = AgglomerativeClustering(linkage='ward', n_clusters=10)
clustering.fit(X_stand)
plt.subplot(223)
plt.scatter(X_stand[:,0], X_stand[:,1], color=cmap((clustering.labels_+.5)/10.), s=s)
plt.title("Ward's method")

clustering.fit(X_set2)
plt.subplot(224)
plt.scatter(X_set2[:,0], X_set2[:,1], color=cmap((clustering.labels_+.5)/10.), s=s)

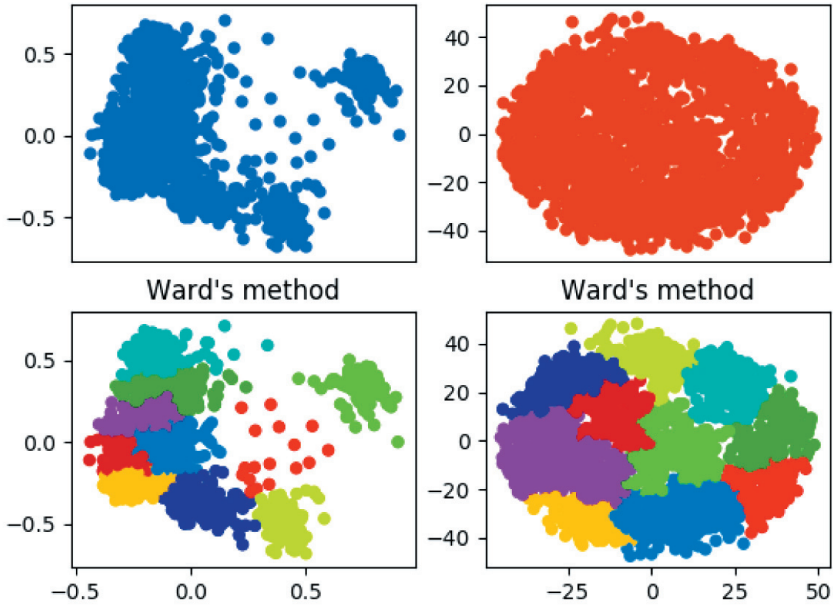
```



```
plt.title("Ward's method")

fig.suptitle("Digits dataset")
plt.show()
```

Digits dataset



```
#####
# Example 5.7.2
# Hierarchical clustering
# Digits dataset, three agglomerative methods, first way of scaling
#####
import numpy as np
from matplotlib import pyplot as plt
from sklearn import manifold, datasets
from sklearn.cluster import AgglomerativeClustering
from sklearn.preprocessing import StandardScaler

digits = datasets.load_digits(n_class=10)
X = digits.data
```

```

y = digits.target
n_samples, n_features = X.shape

np.random.seed(0)
seed = np.random.RandomState(seed=3)
print('be patient the projection on a two-dimensional plane takes a few minutes...\n')
# a projection on a two-dimensional plane it takes a few minutes
X_set = manifold.SpectralEmbedding(n_components=2).fit_transform(X)
X_stand = StandardScaler().fit_transform(X_set)

s=25
plt.close('all')
fig = plt.figure()
cmap = plt.cm.get_cmap('hsv')

plt.subplot(221)
plt.scatter(X_stand[:,0], X_stand[:,1], color=(.2,.4,.7), s=s)
plt.title("Data")
plt.xticks([])

clustering = AgglomerativeClustering(linkage='ward', n_clusters=10)
clustering.fit(X_stand)
plt.subplot(222)
plt.scatter(X_stand[:,0], X_stand[:,1], color=cmap((clustering.labels_+.5)/10.), s=s)
plt.title("Ward's method")
plt.xticks([])

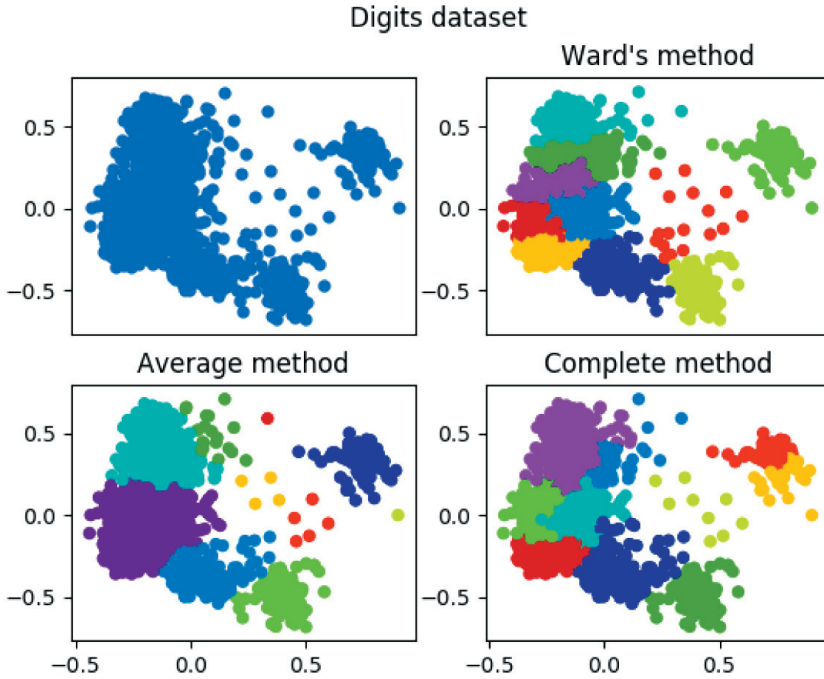
clustering = AgglomerativeClustering(linkage='average', n_clusters=10)
clustering.fit(X_stand)
plt.subplot(223)
plt.scatter(X_stand[:,0], X_stand[:,1], color=cmap((clustering.labels_+.5)/10.), s=s)
plt.title("Average method")

clustering = AgglomerativeClustering(linkage='complete', n_clusters=10)
clustering.fit(X_stand)
plt.subplot(224)

```

```
plt.scatter(X_stand[:,0], X_stand[:,1], color=cmap((clustering.labels_+.5)/10.), s=s)
plt.title("Complete method")
```

```
fig.suptitle("Digits dataset")
plt.show()
```



```
#####
# Example 5.7.3
# Hierarchical clustering
# Digits dataset, three agglomerative methods, second way of scaling
#####
```

```
import numpy as np
from matplotlib import pyplot as plt
from sklearn import manifold, datasets
from sklearn.cluster import AgglomerativeClustering
```

```
digits = datasets.load_digits(n_class=10)
X = digits.data[:1200]
```

```

y = digits.target
n_samples, n_features = X.shape

print('be patient the projection on a two-dimensional plane takes a few minutes...\n')
# a projection on a two-dimensional plane, it takes few minutes
np.random.seed(0)
seed = np.random.RandomState(seed=3)
X_set2 = manifold.MDS(n_components=2,max_iter=3000, eps=1e-9,
                      random_state=seed).fit_transform(X)

s=25
plt.close('all')
fig = plt.figure()
cmap = plt.cm.get_cmap('hsv')

plt.subplot(221)
plt.scatter(X_set2[:,0], X_set2[:,1], color=(.2,.4,.7), s=s)
plt.xticks([])

clustering = AgglomerativeClustering(linkage='ward', n_clusters=10)
clustering.fit(X_set2)
plt.subplot(222)
plt.scatter(X_set2[:,0], X_set2[:,1], color=cmap((clustering.labels_+.5)/10.), s=s)
plt.title("Ward's method")
plt.xticks([])

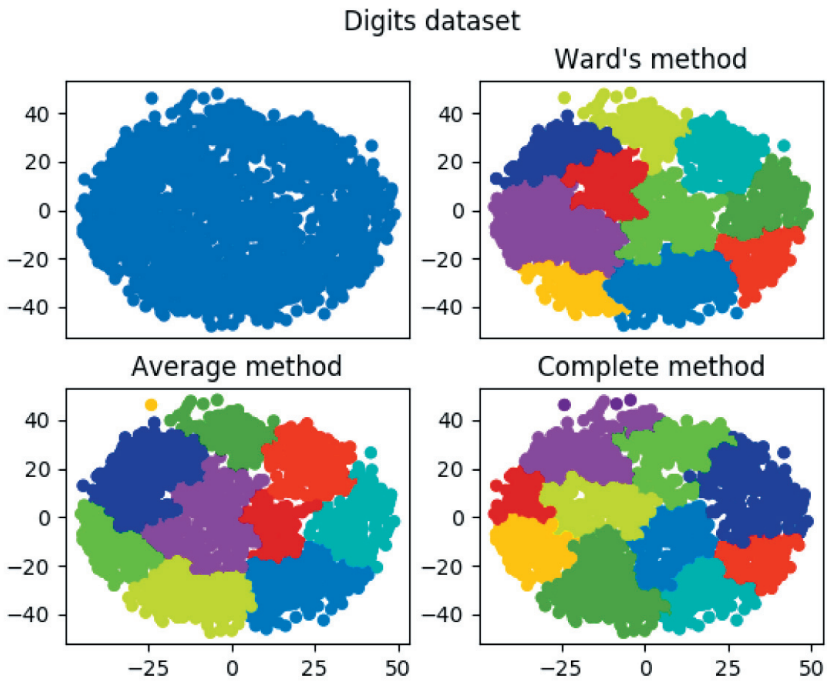
clustering = AgglomerativeClustering(linkage='average', n_clusters=10)
clustering.fit(X_set2)
plt.subplot(223)
plt.scatter(X_set2[:,0], X_set2[:,1], color=cmap((clustering.labels_+.5)/10.), s=s)
plt.title("Average method")

clustering = AgglomerativeClustering(linkage='complete', n_clusters=10)
clustering.fit(X_set2)
plt.subplot(224)

```

```
plt.scatter(X_set2[:,0], X_set2[:,1], color=cmap((clustering.labels_+.5)/10.), s=s)
plt.title("Complete method")
```

```
fig.suptitle("Digits dataset")
plt.show()
```



```
#####
# Example 5.8.1
# Hierarchical clustering
# Iris dataset
#####
```

```
from matplotlib import pyplot as plt
from sklearn import datasets
```

```
# import data
iris = datasets.load_iris()
```

```

X = iris.data[:, :4]
y = iris.target

plt.close('all')
fig = plt.figure()
cmap = plt.cm.get_cmap('hsv')

# Plot the training points
plt.subplot(221)
plt.scatter(X[:, 0], X[:, 1], c=cmap((y+.5)/3.), s=25)
plt.xlabel('Sepal length')
plt.ylabel('Sepal width')
plt.xticks(())
plt.yticks(())

plt.subplot(222)
plt.scatter(X[:, 2], X[:, 3], c=cmap((y+.5)/3.), s=25)
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')
plt.xticks(())
plt.yticks(())

plt.subplot(223)
plt.scatter(X[:, 0], X[:, 2], c=cmap((y+.5)/3.), s=25)
plt.xlabel('Sepal length')
plt.ylabel('Petal Length')
plt.xticks(())
plt.yticks(())

plt.subplot(224)
plt.scatter(X[:, 1], X[:, 3], c=cmap((y+.5)/3.), s=25)
plt.xlabel('Sepal width')
plt.ylabel('Petal Width')
plt.xticks(())

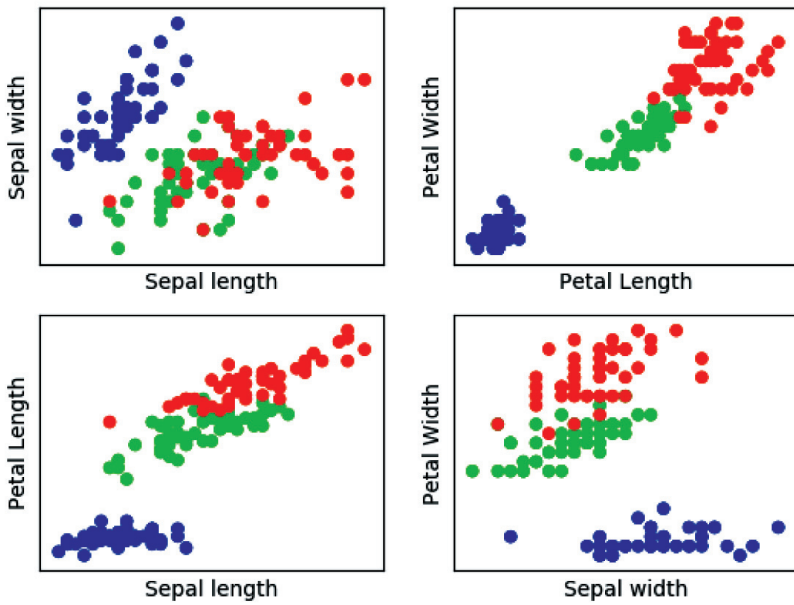
```

```
plt.yticks()
```

```
fig.suptitle("Iris dataset")
```

```
plt.show()
```

Iris dataset



```
#####
```

```
# Example 5.8.2
```

```
# Hierarchical clustering
```

```
# Iris dataset, three methods of clustering, data Petal Length and Petal Width
```

```
#####
```

```
from matplotlib import pyplot as plt
```

```
from sklearn.cluster import AgglomerativeClustering
```

```
from sklearn import datasets
```

```
# import data
```

```
iris = datasets.load_iris()
```

```

X = iris.data[:, 2:]
y = iris.target

plt.close('all')
fig = plt.figure()
cmap = plt.cm.get_cmap('hsv')

# Plot the training points
plt.subplot(221)
plt.scatter(X[:, 0], X[:, 1], c=cmap((y+.5)/3.), s=25)
plt.title("Petal Length vs Petal Width")
plt.xticks(())

# clustering
clustering = AgglomerativeClustering(linkage='ward', n_clusters=3)
clustering.fit(X)
plt.subplot(222)
plt.scatter(X[:,0], X[:,1], color=cmap((clustering.labels_+.5)/3.), s=25)
plt.title("Ward's method")
plt.xticks([])

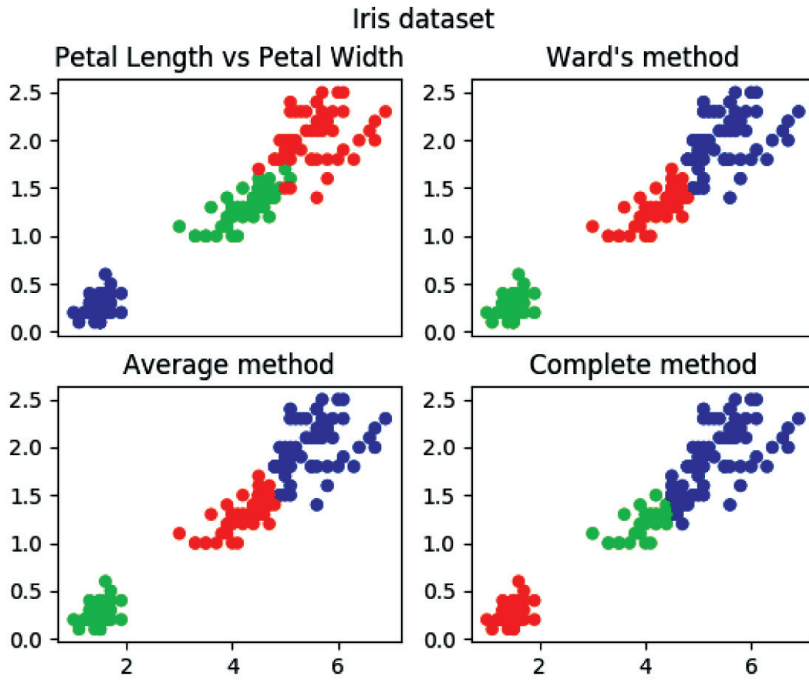
clustering = AgglomerativeClustering(linkage='average', n_clusters=3)
clustering.fit(X)
plt.subplot(223)
plt.scatter(X[:,0], X[:,1], color=cmap((clustering.labels_+.5)/3.), s=25)
plt.title("Average method")

clustering = AgglomerativeClustering(linkage='complete', n_clusters=3)
clustering.fit(X)
plt.subplot(224)
plt.scatter(X[:,0], X[:,1], color=cmap((clustering.labels_+.5)/3.), s=25)
plt.title("Complete method")

```



```
fig.suptitle("Iris dataset")
plt.show()
```



```
#####
# Example 5.8.3
# Hierarchical clustering
# Iris dataset, three method of clustering, data Sepal Length and Petal Width
#####
```

```
from matplotlib import pyplot as plt
from sklearn.cluster import AgglomerativeClustering
from sklearn import datasets
```

```
# import data
iris = datasets.load_iris()
```

```

X = iris.data[:, [0,3]]
y = iris.target

plt.close('all')
fig = plt.figure()
cmap = plt.cm.get_cmap('hsv')

# Plot the training points
plt.subplot(221)
plt.scatter(X[:, 0], X[:, 1], c=cmap((y+.5)/3.), s=25)
plt.title("Sepal Length vs Petal Width")
plt.xticks(())

# clustering
clustering = AgglomerativeClustering(linkage='ward', n_clusters=3)
clustering.fit(X)
plt.subplot(222)
plt.scatter(X[:,0], X[:,1], color=cmap((clustering.labels_+.5)/3.), s=25)
plt.title("Ward's method")
plt.xticks([])

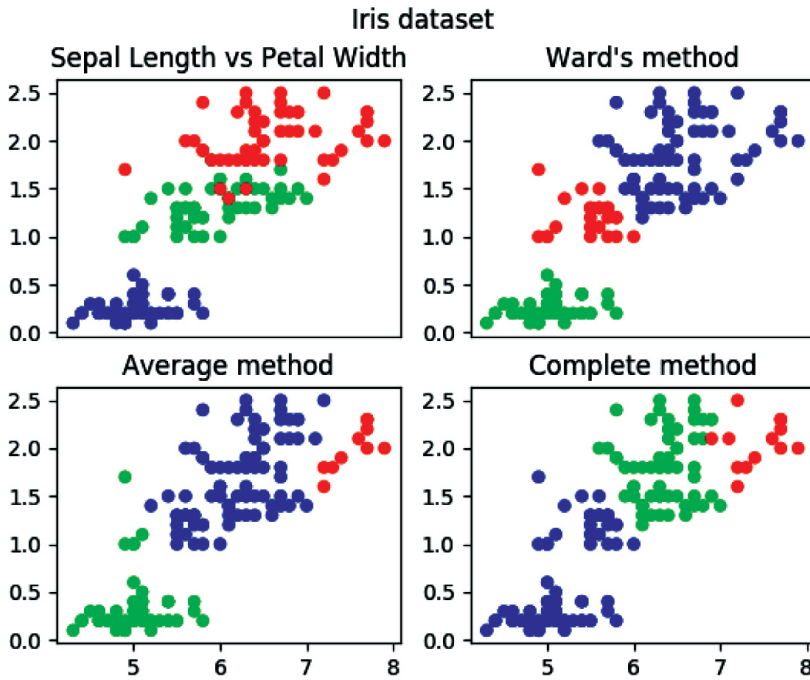
clustering = AgglomerativeClustering(linkage='average', n_clusters=3)
clustering.fit(X)
plt.subplot(223)
plt.scatter(X[:,0], X[:,1], color=cmap((clustering.labels_+.5)/3.), s=25)
plt.title("Average method")

clustering = AgglomerativeClustering(linkage='complete', n_clusters=3)
clustering.fit(X)
plt.subplot(224)
plt.scatter(X[:,0], X[:,1], color=cmap((clustering.labels_+.5)/3.), s=25)
plt.title("Complete method")

```

```
fig.suptitle("Iris dataset")
```

```
plt.show()
```



```
#####
```

```
# Example 5.8.4
```

```
# Hierarchical clustering
```

```
# Iris dataset, three dimensional illustration
```

```
#####
```

```
from matplotlib import pyplot as plt
```

```
from sklearn.cluster import AgglomerativeClustering
```

```
from sklearn import datasets
```

```
from mpl_toolkits.mplot3d import Axes3D
```

```
# import data
```

```

iris = datasets.load_iris()
X = iris.data
y = iris.target

plt.close('all')
number = 1
fig = plt.figure(number)
cmap = plt.cm.get_cmap('hsv')

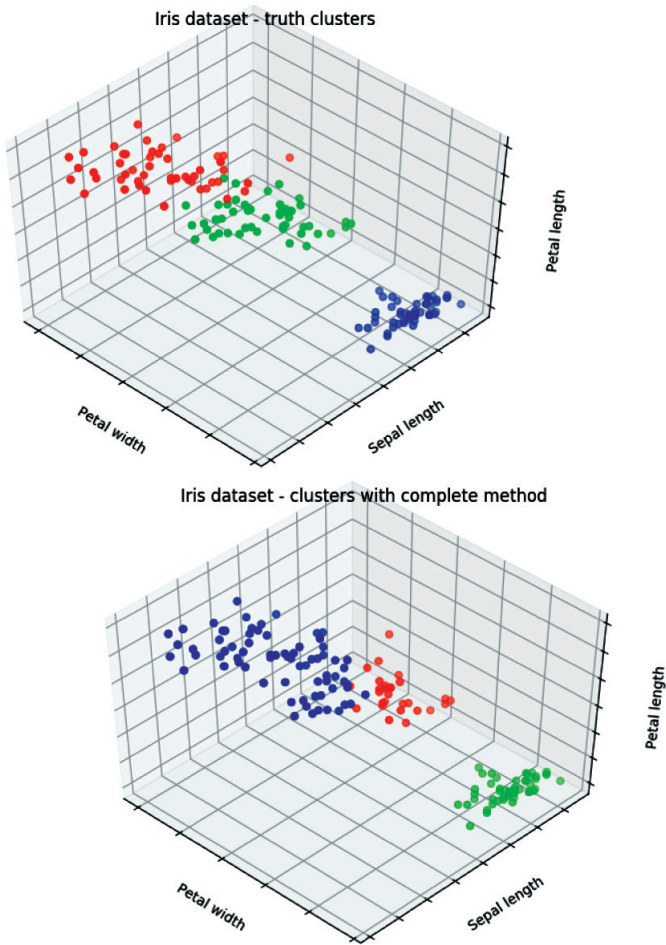
ax = Axes3D(fig, rect=[0, 0,.95, 1], elev=48, azimuth=134)
ax.scatter(X[:, 3], X[:, 0], X[:, 2], color=cmap((y+.5)/3.), s=25)
ax.w_xaxis.set_ticklabels([])
ax.w_yaxis.set_ticklabels([])
ax.w_zaxis.set_ticklabels([])
ax.set_xlabel('Petal width')
ax.set_ylabel('Sepal length')
ax.set_zlabel('Petal length')
fig.suptitle("Iris dataset – truth clusters")

clustering = AgglomerativeClustering(linkage='complete', n_clusters=3)
clustering.fit(X)

number = number + 1
fig = plt.figure(number)
ax1 = Axes3D(fig, rect=[0, 0,.95, 1], elev=48, azimuth=134)
ax1.scatter(X[:, 3], X[:, 0], X[:, 2], color=cmap((clustering.labels_+.5)/3.), s=25)
ax1.w_xaxis.set_ticklabels([])
ax1.w_yaxis.set_ticklabels([])
ax1.w_zaxis.set_ticklabels([])
ax1.set_xlabel('Petal width')
ax1.set_ylabel('Sepal length')
ax1.set_zlabel('Petal length')

```

```
fig.suptitle("Iris dataset – clusters with complete method")
plt.show()
```



```
#####
# Example 5.8.5
# Hierarchical clustering
# Iris dataset, three dimensional illustration
#####
```

```
from matplotlib import pyplot as plt
from sklearn.cluster import AgglomerativeClustering
```

```

from sklearn import datasets
from mpl_toolkits.mplot3d import Axes3D

# import data
iris = datasets.load_iris()
X = iris.data
y = iris.target

plt.close('all')
number = 1
cmap = plt.cm.get_cmap('hsv')

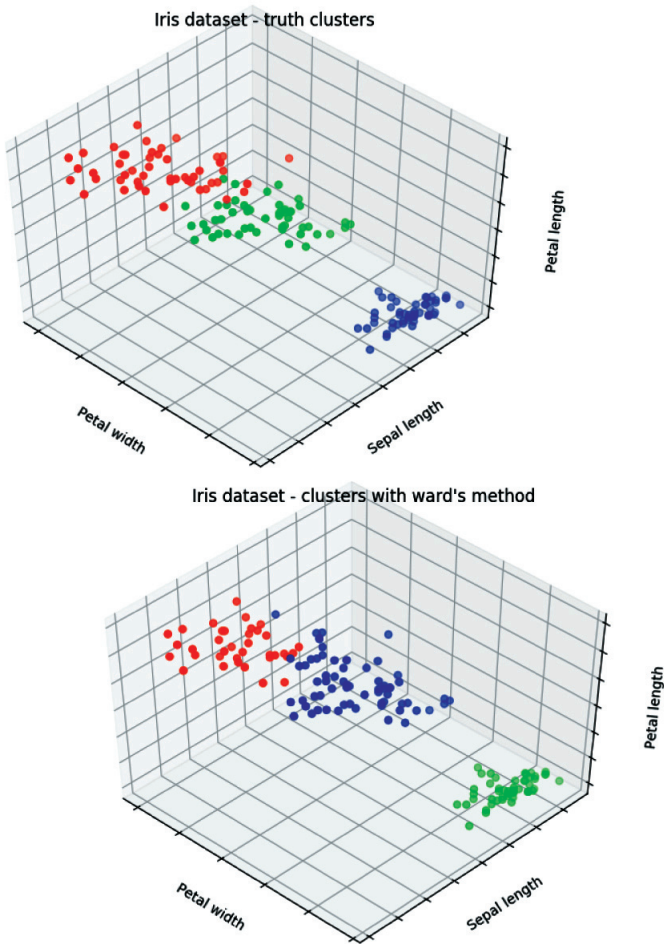
fig = plt.figure(number)
ax = Axes3D(fig, rect=[0, 0,.95, 1], elev=48, azim=134)
ax.scatter(X[:, 3], X[:, 0], X[:, 2], color=cmap((y+.5)/3.), s=25)
ax.w_xaxis.set_ticklabels([])
ax.w_yaxis.set_ticklabels([])
ax.w_zaxis.set_ticklabels([])
ax.set_xlabel('Petal width')
ax.set_ylabel('Sepal length')
ax.set_zlabel('Petal length')
fig.suptitle("Iris dataset – truth clusters")

clustering = AgglomerativeClustering(linkage='ward', n_clusters=3)
clustering.fit(X)

number = number + 1
fig = plt.figure(number)
ax1 = Axes3D(fig, rect=[0, 0,.95, 1], elev=48, azim=134)
ax1.scatter(X[:, 3], X[:, 0], X[:, 2], color=cmap((clustering.labels_+.5)/3.), s=25)
ax1.w_xaxis.set_ticklabels([])
ax1.w_yaxis.set_ticklabels([])
ax1.w_zaxis.set_ticklabels([])

```

```
ax1.set_xlabel('Petal width')
ax1.set_ylabel('Sepal length')
ax1.set_zlabel('Petal length')
fig.suptitle("Iris dataset – clusters with ward's method")
plt.show()
```



## 5.4. Nonhierarchical algorithms

Nonhierarchical algorithms have gained great popularity due to the fact that a given problem of optimization is their cornerstone. In particular, the point is that grouping the initial set of objects in clusters is the solution to some extremum

problem (see in Hand et al. 2001, Amorim, Mirkin 2012, Bock 1999, Diday, Simon 1976, Jain et al. 1999, Jain 2010, Kaufman, Rousseeuw 1990, Mirkin 2005, Rokach 2009, Pedregosa et al. 2011). Let's consider some of the most popular methods.

One of the most popular numerical analyses is the least-squares method. For the problem of the clustering it looks as follows

$$\sum_{j=1}^k \sum_{i=1}^{n_j} |x_i - s_j|^2 \rightarrow \min,$$

on all  $s_j$  and  $k$ .

Numerical implementation of this task is called  $k$ -means method (see Hand et al. 2001, Jain et al. 1999, Jain 2010, Mirkin 2005, Rokach 2009, Sneath, Sokal 1973).

### 5.4.1. $K$ -means method

The idea behind the method is the following:  $k$  of any initial centers gets out of the set in the beginning  $\mathfrak{S}$ . Furthermore, all objects break into  $k$  groups, the closest to the relevant center. On the following step the centers of the found clusters are calculated. The procedure repeats iterative until the centers of clusters are stabilized.

Algorithm of splitting objects  $x_i$  ( $i = 0, 1, \dots, n$ ) is based on minimization of inter-cluster distance. If as such distance the mean square norm  $\ell_2$  is used, the target function is as follows

$$S = \sum_{j=1}^k \sum \left\{ |x_i - \mu_j|^2 \mid x_i \in c_j \right\},$$

where  $x_i$  -  $i$ -th the object, and  $c_j$  represents  $j$ -th cluster with the center  $\mu_j$ .

The structure of the algorithm is as follows:

1. For initialization of the algorithm we randomly choose  $k$  centers of clusters.
2. we put each of  $n$  objects into the cluster, conducting the minimization  $\ell_2$ -norm between the object and the center of the corresponding cluster.
3. We calculate the centers of again received clusters.
4. For each  $i$ -th, if  $x_i \in c_j$  let's calculate

$$h = \operatorname{argmin} \left\{ \frac{n_r \|x_i - \mu_r\|_2}{n_r - 1} \right\},$$

where  $n_r$  number of objects of the cluster  $c_r$ .



For the solution of this task among all elements of the cluster  $x \in c_i$  let's find the element  $z$  minimizing evasion  $\sum_{x \in c_i} \|x - z\|_2^2$  for which we will find the solution of the task:

$$\frac{\partial}{\partial z} \sum_{x \in c_i} \|x - z\|_2^2 = \frac{\partial}{\partial z} \sum_{x \in c_i} (\|x\|_2^2 - 2x^T z + \|z\|_2^2) = \sum_{x \in c_i} (-x + z) = 0,$$

that is:

$$z = \frac{1}{n_i} \sum_{x \in c_i} x.$$

5. If the condition below is satisfied:

$$\frac{n_h \|x_i - \mu_h\|_2}{n_h - 1} < \frac{n_j \|x_i - \mu_j\|_2}{n_j - 1},$$

then the object  $x_i$  should be moved from the cluster  $c_j$  to the cluster  $c_h$  and then again the values of cluster centers should be counted.

6. If  $i < n$ , then we pass to the step 4, otherwise to the step 3.

As a criterion of the stop algorithm one can assume either achievement of the set number of algorithm iterations, or reaching the fixed threshold value by objective function.

The method is effective if data are divided into compact groups which can be described as the sphere. The use of indicator function allows to simplify the notation of the basic algorithm and it can be written down in the following form.

Let  $C = \{c_i\}_{i=1}^k$  be a set of clusters with the centers:

$$\mu_i = \frac{\sum \{x_j \mid x_j \in c_i\}}{\sum \{1 \mid x_j \in c_i\}} = \frac{\sum_{j=1}^n u_j^i x_j}{\sum_{j=1}^n u_j^i},$$

where  $u_j^i$  is an indicator function, that is:

$$u_j^i = \begin{cases} 1, & \text{if } x_j \in c_i, \\ 0, & \text{otherwise.} \end{cases}$$

The objective function

$$S(C, \mathfrak{S}) = \sum_{i=1}^k \sum_{j=1}^n u_j^i d(x_j, \mu_i),$$

and conditions

$$\sum_{i=1}^n u_j^i = 1, \quad 0 < \sum_{j=1}^k u_j^i \leq n,$$

that is, each element can be only in one cluster, and, the cluster cannot be empty. Not to mention that cluster cannot contain more elements than their initial quantity.

The stop condition of algorithm execution after  $v$ -th step can take the following form

$$\left| S^v(C, \mathfrak{S}) - S^{v-1}(C, \mathfrak{S}) \right| < \varepsilon,$$

where  $\varepsilon$  is the chosen threshold.

Let's notice that at the calculation of accessory criterion, it is possible to consider the cluster size that allows to improve efficiency of the algorithm. The criterion is that  $j$ -th element belongs to  $i$ -th, but not  $k$ -th cluster and takes the following form

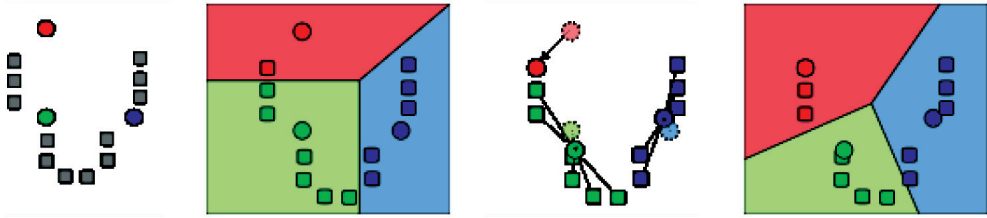
$$\frac{n_i}{n_i - 1} d(x_j, \mu_i) < \frac{n_k}{n_k - 1} d(x_j, \mu_k),$$

where  $n_i$  is the number of elements in the correlated  $c_i$  cluster. The speed of the method convergence is  $O(n)$ .

The literature on the subject mentions the following disadvantages of his method:

- Existence of the priori information on quantity of clusters,
- Sensitivity to the isolated remote elements,
- Essential dependence of speed of the method convergence on the initial choice of the cluster centers.

Let's give the illustration (see Fig. 5.8) of the  $k$ -means method from [http://en.wikipedia.org/wiki/K-means\\_algorithm](http://en.wikipedia.org/wiki/K-means_algorithm).



- 1) Input data (in this case  $K = 3$ ). The starting centers of clusters are chosen randomly and shown in red, blue and green color.
- 2) The clusters are created by merging each center with the nearest mean.
- 3) The barycentre of each of  $K$  clusters becomes the new center.
- 4) Steps 2 and 3 are repeated so far the process of the clustering will not be complete.

**Fig. 5.8.** An illustration of the  $k$ -means method  
 Source: [http://en.wikipedia.org/wiki/K-means\\_algorithm](http://en.wikipedia.org/wiki/K-means_algorithm)

### 5.4.2. Fuzzy $k$ -means

This algorithm is generalization of the mentioned above method. We can use it if clusters are indistinct sets, and, the element can belong to different clusters with different degree of reliability (see Bezdeck 1981, Babuska et al. 2002).

Let  $w \in (1, \infty)$  (usually  $w = 2$  undertakes) the weighting coefficient of the illegibility. Let  $C = \{c_i\}_{i=1}^k$  be a set of clusters with the centers

$$s_i = \frac{\sum_{j=1}^n (u_j^i)^w x_j}{\sum_{j=1}^n (u_j^i)^w},$$

where:

$$u_j^i = \begin{cases} \mu, & \text{if } x_j \in c_i \\ 0, & \text{otherwise,} \end{cases}$$

where  $\mu$  is the membership function.

The objective function

$$S(C, \mathfrak{S}) = \sum_{i=1}^k \sum_{j=1}^n (u_j^i)^w d(x_j, \mu_i).$$

And conditions

$$\sum_{i=1}^n u_j^i = 1, \quad 0 < \sum_{j=1}^k u_j^i \leq n,$$

that is, each element can be only in one cluster, and, the cluster cannot be empty. Moreover the cluster cannot contain more elements than their initial quantity.

The stop condition of the algorithm execution after  $v$ -th step can take the form

$$\left| S^v(C, \mathfrak{S}) - S^{v-1}(C, \mathfrak{S}) \right| < \varepsilon,$$

where  $\varepsilon$  is the chosen threshold. The speed of the method convergence is  $O(n)$ .

### 5.4.3. Gyustafsona–Kessel’s clustering

In fact, it is the same algorithm as mentioned above, but we apply correlation dependences in the clusters. This is the reason that clusters instead of spherical shape become ellipsoids. That allows to carry out splitting with higher quality if elements  $\mathfrak{S}$  are extended along any directions. In other words, if there is a set shape defining the cluster, then it is better to consider elements’ belonging to the cluster according to these shapes.

Well, it is the same  $k$ -means method, but Mahalanobis’s distance is used.

#### FOREL (Formal Element)

This algorithm is one of modifications of  $k$ -means algorithm. Difference consists that the proximity of elements in the cluster is understood as covering the sphere with a fixed radius.

The scheme representing how the algorithm works is shown as follows: the center of the cluster (at the first stage this is any element) is selected, and, all elements whose distance from the center is not greater than the determined  $R$  are assigned to the group. Then the center as the center of gravity (relative to distance) of the received new cluster is recalculated. Group elements are re-verified as described above. And so on until the center of gravity is stabilized (it stops changing).

It is worth noting that the  $k$ -means method, as well as its modifications, is guided a priori by information about number of classes. It can be bypassed. We apply the method  $k$ -means or its modification consistently to each  $k$  clusters and we calculate the maximum mistake. As soon as the value of mistake is stabilized, the number of clusters elements is settled then the algorithm stops. The method is long.

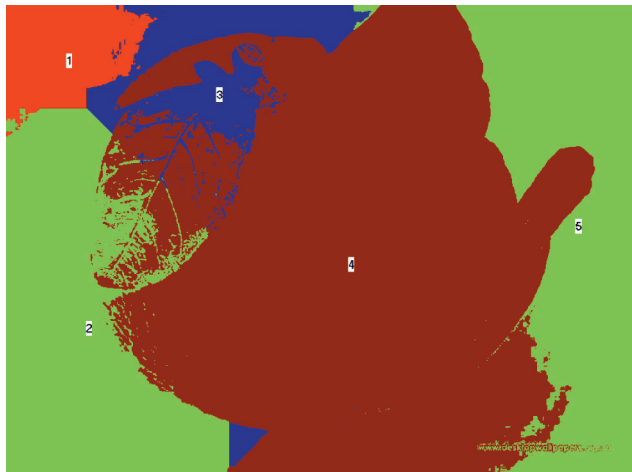
Let's have the illustration representing the use of the  $k$ -means method for the clustering of images. So, let's look at the test of image "Orange" (see Fig. 5.9).



**Fig. 5.9.** The image "Orange"

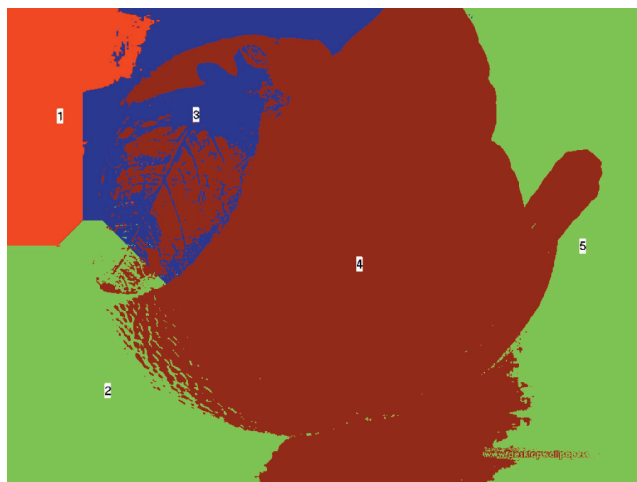
Let's consider the  $k$ -means method for the array  $(i, j, r_{i,j}, g_{i,j}, b_{i,j})$ . We choose quantity of clusters equal to five.

After the first step we receive the following clusters (see Fig. 5.10).



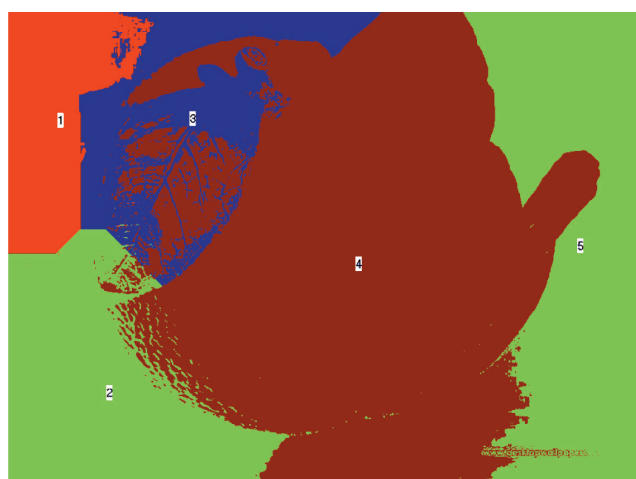
**Fig. 5.10.** The received, after first step, clusters from the image "Orange"

After the sixth step clusters are distributed as it is shown in Figure 5.11.



**Fig. 5.11.** The received, after sixth step, clusters from the imagine “Orange”

The picture does not change significantly even after hundred steps (see Fig. 5.12).



**Fig. 5.12.** The received, after hundred steps, clusters from the imagine “Orange”

#### **5.4.4. Method of correlation galaxies**

Undoubtedly, the idea of creating clusters, involving maximizing correlation communication is the cornerstone of this method, i.e. the sum of correlation factors modules between parameters of one group which is rather big, and communication between parameters from different groups which are small. On the complete objects

correlation matrix the graph which then breaks into subgraphs is constructed. Moreover, elements corresponding to each of subgraphs also form the cluster.

Let's consider the complete correlation matrix  $C = \{c_{i,j}\}$ ,  $i, j = 1, 2, \dots, n$ . We will order the initial objects in a way based on the principle of the maximum correlation. Thus in total  $n$  of objects joint by  $n - 1$  of lines (edges) the sum of modules of correlation factors is maximum. Let's describe creation of the graph. At the beginning we can consider the tops of the graph corresponding to objects  $x_l$  and  $x_m$ . We will deliver these tops in compliance weight  $c^{(1)} = |c_{l,m}|$  to the edge connecting. Then, having excluded  $c_{l,m}$ , we find the greatest coefficient in the  $m$ -th matrix column (it corresponds with finding of the element, which is later the most related  $x_l$  "is connected" with  $x_m$ ). And the greatest coefficient in the  $l$ -th matrix row (it corresponds with finding of the element, which is later the most related  $x_m$  "is connected" with  $x_l$ ). The greater of two indicated factors is chosen. Let it be  $c^{(2)} = |c_{l,m}|$ . The vertex  $x_j$  we connect with  $x_l$ , and, to the corresponding edge we put down value  $c^{(2)}$ . Then we find objects, the most connected with  $x_l$ ,  $x_m$  and  $x_j$ , and we choose the greatest of the found correlation factors. Let it be  $c^{(3)} = |c_{j,q}|$ . Let's have a new element, appearing at every stage as already used, be excluded. Therefore,  $q \neq l$ ,  $q \neq m$ ,  $q \neq j$ . Furthermore, the top of the graph corresponding  $x_q$ , is connected with  $x_j$ , etc. At each step the elements, which are most strongly connected with two last considered elements, are defined. Then one of them corresponding to a larger correlation factor is removed. The procedure comes to an end after  $n - 1$  steps. The graph consists of the vertexes connected by  $n - 1 - m$  edges. Next we set threshold value  $\epsilon$ . Thus all edges corresponding to the less than the fixed  $\epsilon$ , as a correlation factors, are excluded from the graph. The received subgraphs create clusters.

### 5.4.5. Spectral clustering method

The classification task is to determine disjoint subsets in such a way that the similarity of objects within subsets is as high as possible, but between subsets as low as possible. The basis of such research is the assumption that data obtained from one source should behave similarly. Sometimes similarity is very hard defining. The method spectral clustering is dedicated for extracting non-convex boundaries groups.

Let us assume that objects  $x_i$  ( $i = 0, 1, \dots, n$ ) are from  $m$ -dimensional space and form not compact but connectivity clusters. Samples input data are presented in Figure 5.13. Spectral clustering makes use of the spectrum (eigenvalues) relating to the Laplace (similarity) matrix to reduce primary  $n$ -dimensional data into a data set of dimension equal to the number of classes  $k$ .

A discussed method (see for example Chan et al. 1994, Dhillon et al. 2004, Jenssen et al. 2004, Kannan et al. 2000, von Luxburg et al. 2008, von Luxburg 2007, Ng, Jordan,

Weiss 2002, Weiss 1999, Zelnik-Manor, Perona 2004) enables us to detect untypical (as it was mention above not compact and non-convex boundaries, but connectivity) subsets of data. The idea of these algorithms is to cluster points using eigenvectors of matrices derived from the data. The following stages can be distinguished.

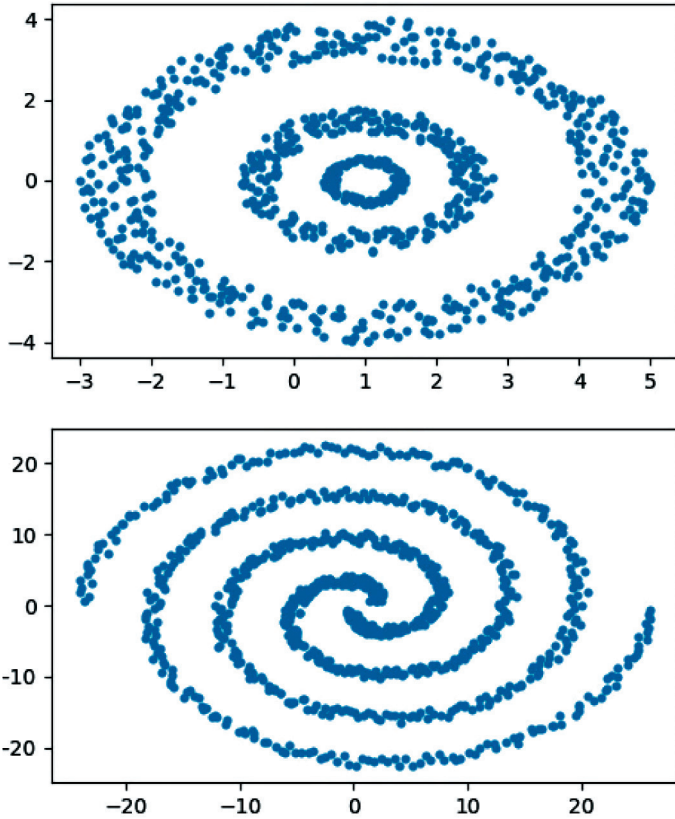


Fig. 5.13. Samples input data for spectral clustering

First step: Let  $X$  be a matrix of data which describes the set of  $n$  elements with  $m$  attributes. Based on this matrix, the distance matrix  $D = [d_{ij}]$  is computed.

Second step: This stage is based on the distance matrix. A similarity (affinity) matrix  $A$  is computed by applying a kernel estimator. In the subject literature, the following kernel estimators can be found: linear, polynomial, hyperbolic, chain, Gaussian (eq. (5.1)) and Laplace (eq. (5.2)). More kernel estimators are given in Hofmann et al. 2008, Langone et al. 2016, Scholkopf et al. 1998, Vapnik 2000, 1998. Furthermore, the type of kernel function to utilize is application-dependent. Table 5.1 outline some of them.



**Table 5.1**  
Types of kernel function for differential application

Application	Kernel name	Mathematical expression
Vector data	Radial Basis Function	$\exp\left(-\frac{d_{ij}^2}{\sigma^2}\right)$
Images	Radial Basis Function $\chi^2$	$\exp\left(-\frac{\chi_{ij}^2}{\sigma^2}\right)$
Text	cosine	$\frac{x_i^T x_j}{\ x_i\  \ x_j\ }$
Time series	Radial Basis Function /correlation distance	$\exp\left(-\frac{d_{ij}^2}{\sigma^2}\right)$ where $d_{ij}$ is a correlation distance

Source: (Langone, Mall, Alzate, Suykens 2016)

The two most important are:

$$A_{ij} = \exp\left(-\frac{d_{ij}^2}{\sigma^2}\right) \tag{5.1}$$

$$A_{ij} = \exp(-\sigma \cdot d_{ij}) \tag{5.2}$$

where  $\sigma$  is a scale parameter called the kernel width. It has a fundamental meaning for spectral classification. In the literature a lot of different methods of determining this parameter can be found (see Zelnik-Manor, Perona 2005).

Third step: The stage consists of the constructing a normalized Laplace matrix  $L = W^{-1/2} \cdot A \cdot W^{1/2}$ , where  $W$  is a diagonal matrix of weights with the diagonal elements being equal to the sum of the row elements of the matrix  $A$ .

Fourth step: Eigenvalues and eigenvectors of the matrix  $L$  are computed. The first  $k$  eigenvectors (corresponding to ordered eigenvalues and chosen to be orthogonal to each other in a case of repeated eigenvalues) are used to build the  $n \times k$ -matrix  $E$ .

Fifth step: In the last stage, the matrix  $E$  is normalized so that the length of each row is one (i.e.  $E'_{ij} = E_{ij} / (\sum_j E_{ij}^2)^{1/2}$ ).

Sixth step: Thus obtained matrix  $E'$  is a starting point for classical clustering method (e.g.,  $k$ -mean method).

The most important problem is to depict number of clusters. There are several criterion discussed in for instance Chan et al. 1994, Jensen et al. 2004, Kannan et al. 2000, Zelnik-Manor, Perona 2004. The reader can delve into the subject oneself.

Let us show the two examples. The data used in the first case are shown in Figure 5.14. They form three circles with a common circle center. It is obvious that in this case we can distinguish three subsets.

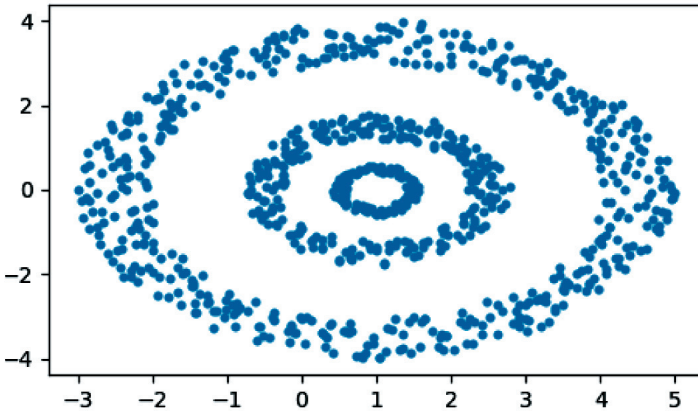


Fig. 5.14. The first data set used for spectral clustering – noisy circles

Such cluster can be easy identified by proposed transforming data. In such case the Laplacian  $L$  from the third step is approximately block-diagonal, with each block defining a cluster. It can be represented as it is shown in Figure 5.15. The result of data clustering is illustrated in Figure 5.16.

$$L = \begin{bmatrix} L_1 & & \\ & L_2 & \\ & & L_3 \\ & 0 & & 0 \end{bmatrix}$$

Fig. 5.15. The Laplacian matrix for the data shown in Figure 5.14

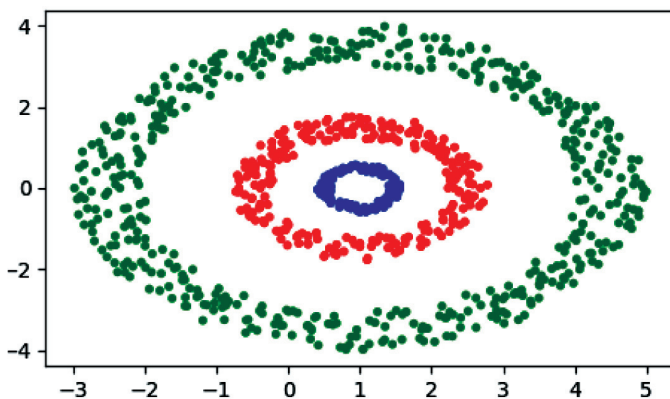


Fig. 5.16. Depicted clusters from prepared data

In the second example the data are called spirals (see in Fig. 5.17). There are two good separated clusters. More examples the reader can find in code programmers on the web page.

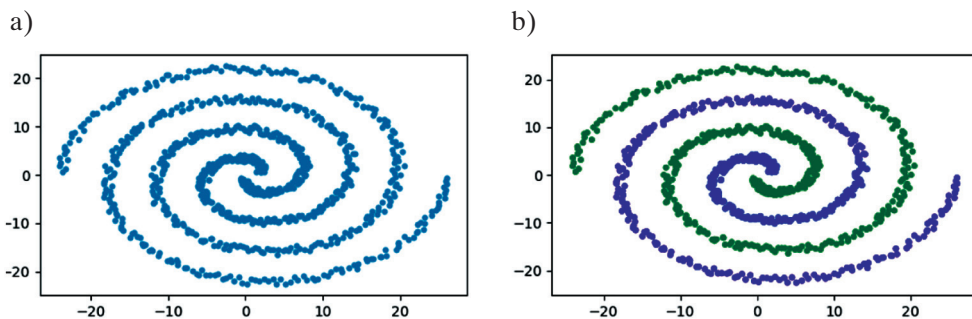


Fig. 5.17. The 'spirals' data set: a) source data set; b) depicted clusters

## 5.5. Examples in Python – clustering nonhierarchical methods

```
#####
# Example 5.9.1
# k-means clustering
# Dataset from table 2.2
#####
import numpy as np
import pandas as pd
```

```

import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score

# read data – data from table 2.2
x = [1,2,3,4,5,6,7,8]
y = [2,3,2,4,4,7,6,7]
# data normalization
x1 = (x - np.mean(x))/np.std(x)
y1 = (y - np.mean(y))/np.std(y)

dane = pd.DataFrame({
    'x1': np.array(x1),
    'y1': np.array(y1),
})
range_n_clusters = [2, 3, 4]

plt.close('all')
cmap = plt.cm.get_cmap('Dark2')

for n_clusters in range_n_clusters:
    # Create a subplot with 1 row and 2 columns
    fig, (ax1, ax2) = plt.subplots(1, 2)
    fig.set_size_inches(10, 5)

    # k-means method
    clusterer = KMeans(n_clusters=n_clusters, random_state=10)
    cluster_labels = clusterer.fit_predict(dane)
    # Compute the average value of silhouette
    silhouette_avg = silhouette_score(dane, cluster_labels)
    print("For n_clusters =", n_clusters,
          "The average silhouette_score is :", silhouette_avg)

    # 1st Plot showing the data
    ax1.scatter(dane.x1, dane.y1, c=cmap( [(1 +.3 )/ n_clusters]*8 ))
    ax1.set_title("The data")
    # 2nd Plot showing the actual clusters formed
    colors = cmap((cluster_labels.astype(float) +.3) / n_clusters)
    ax2.scatter(dane.x1, dane.y1, c=colors)

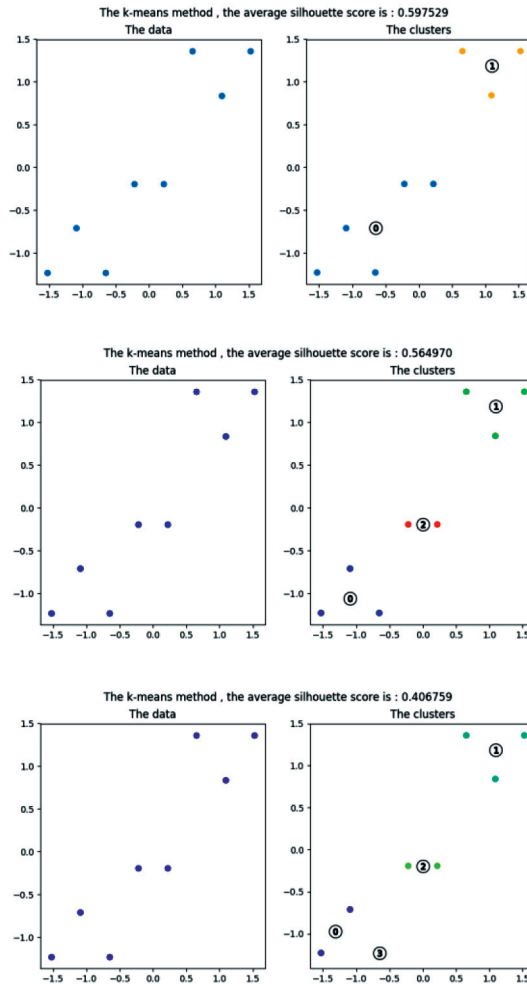
```

```

ax2.set_title("The clusters")
# Labeling the clusters
centers = clusterer.cluster_centers_
# Draw white circles at cluster centers
ax2.scatter(centers[:, 0], centers[:, 1], marker='o',
            c="white", alpha=1, s=200, edgecolor='k')
for i, c in enumerate(centers):
    ax2.scatter(c[0], c[1], marker='$%d$' % i, alpha=1,
                s=50, edgecolor='k')
fig.suptitle("The k-means method, the average silhouette score is : %f " % silhouette_avg)

plt.show()

```



```

#####
# Example 5.9.2
# k-means clustering
# Masterpieces dataset (from B.Mirkin Clustering for data mining)
#####

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score

x = [19, 29.4, 23.9, 18.4, 25.7, 12.1, 23.9, 27.2]
y = [43.7, 36, 38, 27.9, 22.3, 16.9, 30.2, 58]
# plt.scatter(x, y)
x1 = (x - np.mean(x)) / np.std(x)
y1 = (y - np.mean(y)) / np.std(y)

dane = pd.DataFrame({
    'x1': np.array(x1),
    'y1': np.array(y1),
})

range_n_clusters = [2, 3, 4]
cmap = plt.cm.get_cmap('Dark2')
plt.close('all')
for n_clusters in range_n_clusters:
    # Create a subplot with 1 row and 2 columns
    fig, (ax1, ax2) = plt.subplots(1, 2)
    fig.set_size_inches(10, 5)

    # k-means method
    clusterer = KMeans(n_clusters=n_clusters, random_state=10)
    cluster_labels = clusterer.fit_predict(dane)
    # Compute the average value of silhouette
    silhouette_avg = silhouette_score(dane, cluster_labels)
    print("For n_clusters =", n_clusters,
          "The average silhouette_score is :", silhouette_avg)

    # 1st Plot showing the data
    ax1.scatter(dane.x1, dane.y1, c=cmap( [(1 +.3) / n_clusters]*8 ))
    ax1.set_title("The data")
    # 2nd Plot showing the actual clusters formed

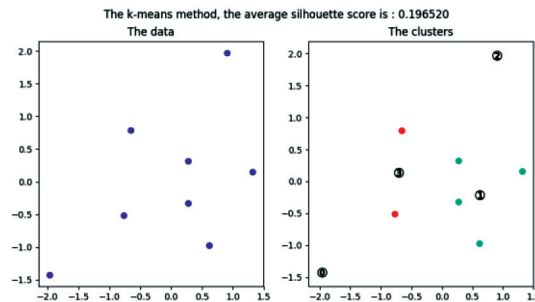
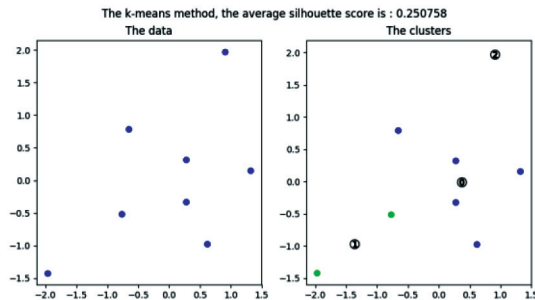
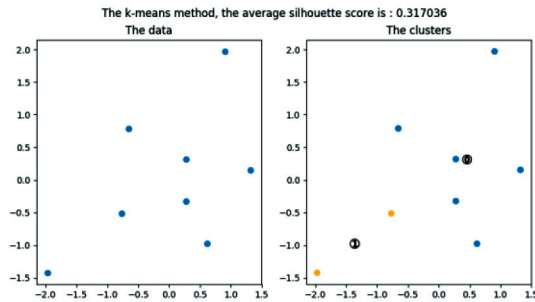
```

```

colors = cmap((cluster_labels.astype(float) +.7) / n_clusters)
ax2.scatter(dane.x1, dane.y1, c=colors)
ax2.set_title("The clusters")
# Labeling the clusters
centers = clusterer.cluster_centers_
# Draw white circles at cluster centers
ax2.scatter(centers[:, 0], centers[:, 1], marker='o',
           c="white", alpha=1, s=100, edgecolor='k')
for i, c in enumerate(centers):
    ax2.scatter(c[0], c[1], marker='$_d$' % i, alpha=1,
               s=50, edgecolor='k')
fig.suptitle("The k-means method, the average silhouette score is : %f " % silhouette_avg)

plt.show()

```



```

#####
# Example 5.10
# k-means clustering
# Generating dataset (blobs)
#####
import numpy as np
import matplotlib.pyplot as plt

from sklearn import datasets
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score

np.random.seed(0)
# The size should be big enough to see the scalability of the algorithms,
# but not too big to avoid too long running times

n_samples = 2500
n_features = 2
centers = [(-5, -5), (-4, -2), (5, 5)]

blobs = datasets.make_blobs(n_samples=n_samples, n_features=n_features, cluster_std=1.0,
                           centers=centers, shuffle=False, random_state= 120)

transformation = [[ 0.60834549, -0.63667341], [-0.40887718, 0.85253229]]
X, y = blobs
X_flattened_blobs = np.dot(X, transformation)

range_n_clusters = [2, 3, 4, 5, 6, 7, 8]
cmap = plt.cm.get_cmap('Dark2')

plt.close('all')
for n_clusters in range_n_clusters:
    # Create a subplot with 1 row and 2 columns
    fig, (ax1, ax2) = plt.subplots(1, 2)
    fig.set_size_inches(10, 5)

    # k-means method
    clusterer = KMeans(n_clusters=n_clusters, random_state=10)

```



```

X_stand = StandardScaler().fit_transform(X)
cluster_labels = clusterer.fit_predict(X_stand)
# Compute the average value of silhouette
silhouette_avg1 = silhouette_score(X_stand, cluster_labels)
print("For n_clusters =", n_clusters, "blobs",
      "The average silhouette_score is :", silhouette_avg1)
# 2nd Plot showing the actual clusters formed
colors = cmap((cluster_labels.astype(float) +.5) / n_clusters)
ax1.scatter(X_stand[:,0], X_stand[:,1], c=colors)
ax1.set_title("The clusters of dataset blobs")
# Labeling the clusters
centers = clusterer.cluster_centers_
# Draw white circles at cluster centers
ax1.scatter(centers[:, 0], centers[:, 1], marker='o',
            c="white", alpha=1, s=100, edgecolor='k')
for i, c in enumerate(centers):
    ax1.scatter(c[0], c[1], marker='%d$' % i, alpha=1,
               s=50, edgecolor='k')

```

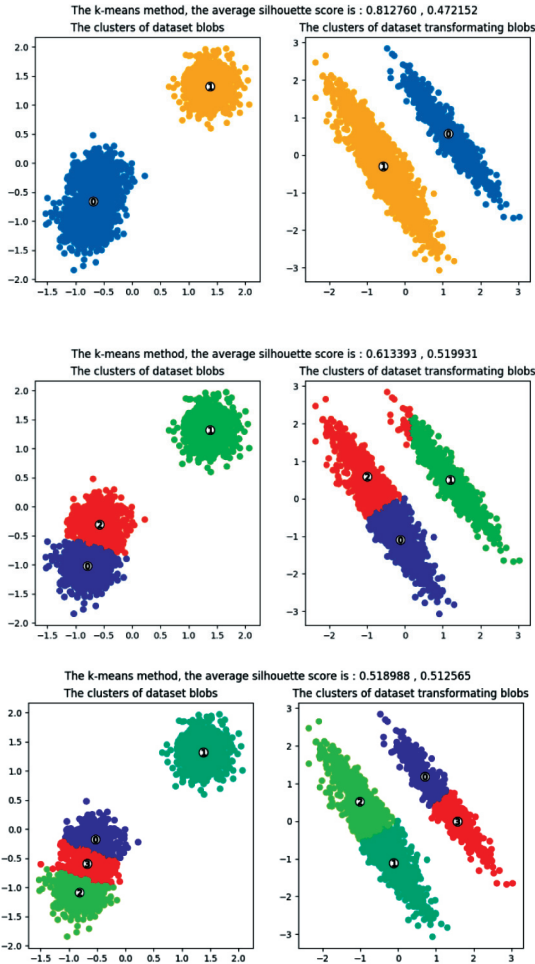
```

X_stand = StandardScaler().fit_transform(X_flattened_blobs)
cluster_labels = clusterer.fit_predict(X_stand)
# Compute the average value of silhouette
silhouette_avg2 = silhouette_score(X_stand, cluster_labels)
print("For n_clusters =", n_clusters, "transformating blobs",
      "The average silhouette_score is :", silhouette_avg2)
# 2nd Plot showing the actual clusters formed
colors = cmap((cluster_labels.astype(float) +.5) / n_clusters)
ax2.scatter(X_stand[:, 0], X_stand[:, 1], c=colors)
ax2.set_title("The clusters of dataset transformating blobs")
# Labeling the clusters
centers = clusterer.cluster_centers_
# Draw white circles at cluster centers
ax2.scatter(centers[:, 0], centers[:, 1], marker='o',
            c="white", alpha=1, s=100, edgecolor='k')
for i, c in enumerate(centers):

```

```
ax2.scatter(c[0], c[1], marker='$%d$' % i, alpha=1,
            s=50, edgecolor='k')
fig.suptitle("The k-means method, the average silhouette score is : %f, %f" % (silhouette_
avg1,silhouette_avg2))
```

```
plt.show()
```



```
#####
# Example 5.11
# k-means clustering
# Generating dataset (blobs with different variances)
#####
import numpy as np
import matplotlib.pyplot as plt
```

```

from sklearn import datasets
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score

np.random.seed(0)
# The size should be big enough to see the scalability of the algorithms,
# but not too big to avoid too long running times

n_samples = 2500
n_features = 2
centers = [(-5, -5), (-4, 2), (5, 5)]

blobs = datasets.make_blobs(n_samples=n_samples, n_features=n_features, cluster_std=
    [1.0, 0.5, 2.5],
    centers=centers, shuffle=False, random_state= 120)

transformation = [[ 0.60834549, -0.63667341], [-0.40887718, 0.85253229]]
X, y = blobs
X_flattened_blobs = np.dot(X, transformation)

range_n_clusters = [2, 3, 4, 5, 6]
cmap = plt.cm.get_cmap('Dark2')

plt.close('all')
for n_clusters in range_n_clusters:
    # Create a subplot with 1 row and 2 columns
    fig, (ax1, ax2) = plt.subplots(1, 2)
    fig.set_size_inches(10, 5)

    # k-means method
    clusterer = KMeans(n_clusters=n_clusters, random_state=10)
    X_stand = StandardScaler().fit_transform(X)
    cluster_labels = clusterer.fit_predict(X_stand)
    # Compute the average value of silhouette

```

```

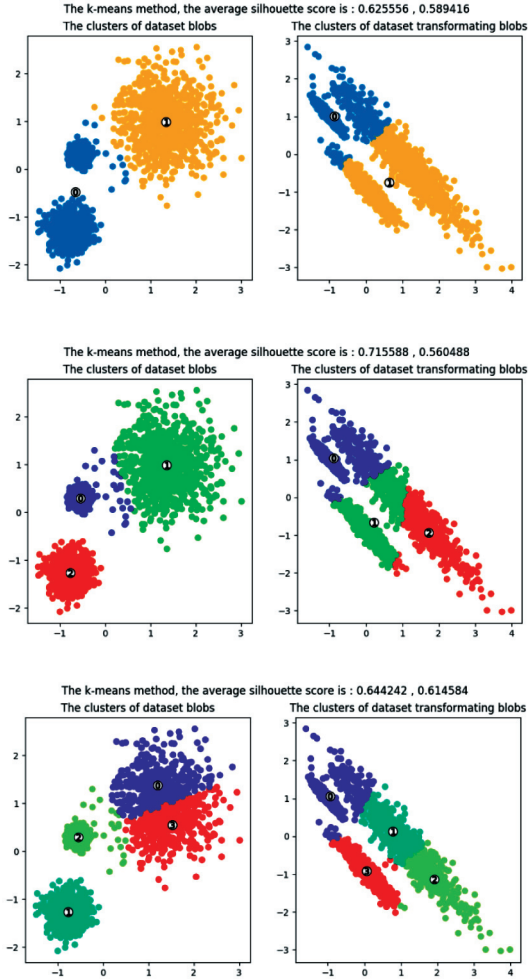
silhouette_avg1 = silhouette_score(X_stand, cluster_labels)
print("For n_clusters =", n_clusters, "blobs",
      "The average silhouette_score is :", silhouette_avg1)
# 2nd Plot showing the actual clusters formed
colors = cmap((cluster_labels.astype(float) +.5) / n_clusters)
ax1.scatter(X_stand[:,0], X_stand[:,1], c=colors)
ax1.set_title("The clusters of dataset blobs")
# Labeling the clusters
centers = clusterer.cluster_centers_
# Draw white circles at cluster centers
ax1.scatter(centers[:, 0], centers[:, 1], marker='o',
            c="white", alpha=1, s=100, edgecolor='k')
for i, c in enumerate(centers):
    ax1.scatter(c[0], c[1], marker='$_d$' % i, alpha=1,
               s=50, edgecolor='k')

X_stand = StandardScaler().fit_transform(X_flattened_blobs)
cluster_labels = clusterer.fit_predict(X_stand)
# Compute the average value of silhouette
silhouette_avg2 = silhouette_score(X_stand, cluster_labels)
print("For n_clusters =", n_clusters, "transforming blobs",
      "The average silhouette_score is :", silhouette_avg2)
# 2nd Plot showing the actual clusters formed
colors = cmap((cluster_labels.astype(float) +.5) / n_clusters)
ax2.scatter(X_stand[:, 0], X_stand[:, 1], c=colors)
ax2.set_title("The clusters of dataset transforming blobs")
# Labeling the clusters
centers = clusterer.cluster_centers_
# Draw white circles at cluster centers
ax2.scatter(centers[:, 0], centers[:, 1], marker='o',
            c="white", alpha=1, s=100, edgecolor='k')
for i, c in enumerate(centers):
    ax2.scatter(c[0], c[1], marker='$_d$' % i, alpha=1,
               s=50, edgecolor='k')

```

```
fig.suptitle("The k-means method, the average silhouette score is : %f, %f" % (silhouette_
avg1,silhouette_avg2))
```

```
plt.show()
```



```
#####
```

```
# Example 5.12
```

```
# k-means clustering
```

```
# Generating dataset (unevenly sized blobs with different variance)
```

```
#####
```

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

```

from sklearn import datasets
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score

n_samples = 2500
n_features = 2
centers = [(-5, -5), (-4, 2), (5, 5)]

blobs = datasets.make_blobs(n_samples=n_samples, n_features=n_features, cluster_std=
    [1.0, 0.5, 2.5],
    centers=centers, shuffle=False, random_state= 120)

transformation = [[ 0.60834549, -0.63667341], [-0.40887718, 0.85253229]]

X, y = blobs
X_filtered = np.vstack((X[y == 0][:500], X[y == 1][:25], X[y == 2][:100]))
X_flattened_blobs = np.dot(X_filtered, transformation)

range_n_clusters = [2, 3, 4, 5, 6]
cmap = plt.cm.get_cmap('Dark2')

plt.close('all')
for n_clusters in range_n_clusters:
    # Create a subplot with 1 row and 2 columns
    fig, (ax1, ax2) = plt.subplots(1, 2)
    fig.set_size_inches(10, 5)

    # k-means method
    clusterer = KMeans(n_clusters=n_clusters, random_state=10)
    X_stand = StandardScaler().fit_transform(X_filtered)
    cluster_labels = clusterer.fit_predict(X_stand)
    # Compute the average value of silhouette
    silhouette_avg1 = silhouette_score(X_stand, cluster_labels)

```

```

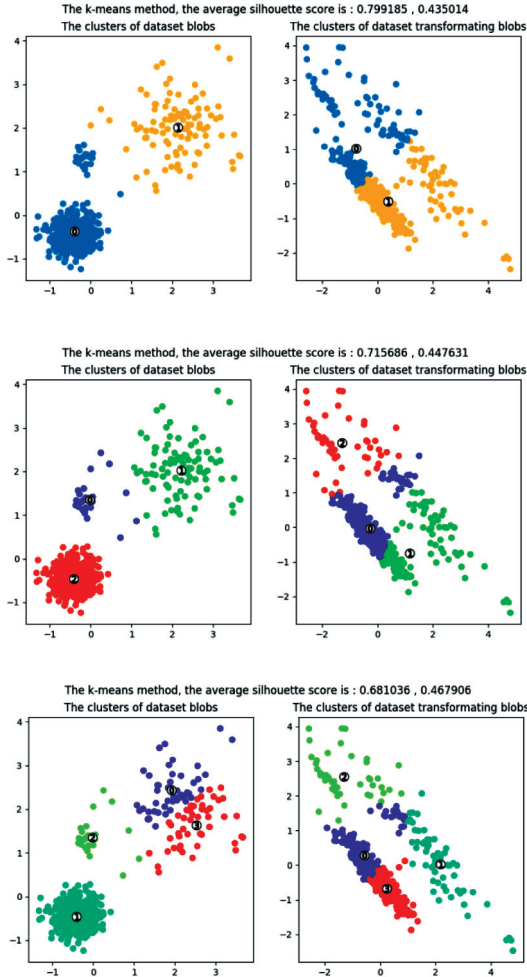
print("For n_clusters =", n_clusters, "blobs",
      "The average silhouette_score is :", silhouette_avg1)
# 2nd Plot showing the actual clusters formed
colors = cmap((cluster_labels.astype(float) +.5) / n_clusters)
ax1.scatter(X_stand[:,0], X_stand[:,1], c=colors)
ax1.set_title("The clusters of dataset blobs")
# Labeling the clusters
centers = clusterer.cluster_centers_
# Draw white circles at cluster centers
ax1.scatter(centers[:, 0], centers[:, 1], marker='o',
            c="white", alpha=1, s=100, edgecolor='k')
for i, c in enumerate(centers):
    ax1.scatter(c[0], c[1], marker='%d$' % i, alpha=1,
               s=50, edgecolor='k')

X_stand = StandardScaler().fit_transform(X_flattened_blobs)
cluster_labels = clusterer.fit_predict(X_stand)
# Compute the average value of silhouette
silhouette_avg2 = silhouette_score(X_stand, cluster_labels)
print("For n_clusters =", n_clusters, "transforming blobs",
      "The average silhouette_score is :", silhouette_avg2)
# 2nd Plot showing the actual clusters formed
colors = cmap((cluster_labels.astype(float) +.5) / n_clusters)
ax2.scatter(X_stand[:, 0], X_stand[:, 1], c=colors)
ax2.set_title("The clusters of dataset transforming blobs")
# Labeling the clusters
centers = clusterer.cluster_centers_
# Draw white circles at cluster centers
ax2.scatter(centers[:, 0], centers[:, 1], marker='o',
            c="white", alpha=1, s=100, edgecolor='k')
for i, c in enumerate(centers):
    ax2.scatter(c[0], c[1], marker='%d$' % i, alpha=1,
               s=50, edgecolor='k')

```

```
fig.suptitle("The k-means method, the average silhouette score is : %f, %f" % (silhouette_
avg1,silhouette_avg2))
```

```
plt.show()
```



```
#####
# Example 5.13
# k-means clustering
# Generating dataset (no structure)
#####
import numpy as np
import matplotlib.pyplot as plt
```



```

from sklearn import datasets
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score

np.random.seed(10)

n_samples = 2500
n_features = 2

range_n_clusters = [2, 3, 4, 5, 6]
X = np.random.rand(n_samples, 2)
X_stand = StandardScaler().fit_transform(X)
cmap = plt.cm.get_cmap('Dark2')

plt.close('all')
for n_clusters in range_n_clusters:
    fig = plt.figure()
    # k-means method
    clusterer = KMeans(n_clusters=n_clusters, random_state=10)
    cluster_labels = clusterer.fit_predict(X_stand)
    # Compute the average value of silhouette
    silhouette_avg = silhouette_score(X_stand, cluster_labels)
    print("For n_clusters =", n_clusters, "no structure",
          "The average silhouette_score is :", silhouette_avg)
    # the Plot showing the actual clusters formed
    colors = cmap((cluster_labels.astype(float) +.5) / n_clusters)
    plt.scatter(X_stand[:, 0], X_stand[:, 1], c=colors)
    plt.title("The clusters of dataset no structure")
    # Labeling the clusters
    centers = clusterer.cluster_centers_
    # Draw white circles at cluster centers
    plt.scatter(centers[:, 0], centers[:, 1], marker='o',
               c="white", alpha=1, s=100, edgecolor='k')

```

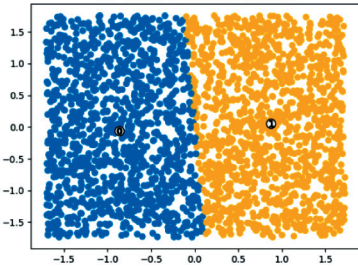
```

for i, c in enumerate(centers):
    plt.scatter(c[0], c[1], marker='$%d$' % i, alpha=1,
                s=50, edgecolor='k')
fig.suptitle("The k-means method, the average silhouette score is : %f "
            % (silhouette_avg))

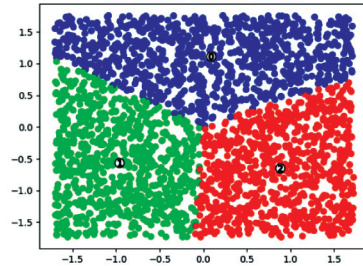
plt.show()

```

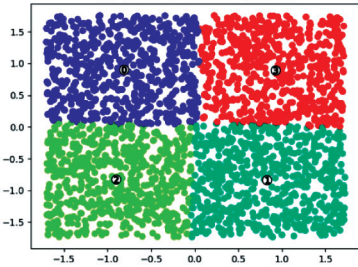
The k-means method, the average silhouette score is : 0.356534  
The clusters of dataset no structure



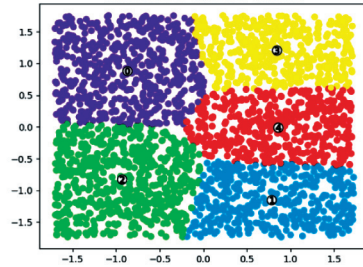
The k-means method, the average silhouette score is : 0.387344  
The clusters of dataset no structure



The k-means method, the average silhouette score is : 0.406232  
The clusters of dataset no structure



The k-means method, the average silhouette score is : 0.379227  
The clusters of dataset no structure



#####

# Example 5.14

# k-means clustering

# Generating dataset (noisy circles)

#####

import numpy as np

import matplotlib.pyplot as plt

from sklearn import datasets

from sklearn.preprocessing import StandardScaler

from sklearn.cluster import KMeans

```

from sklearn.metrics import silhouette_score

np.random.seed(10)

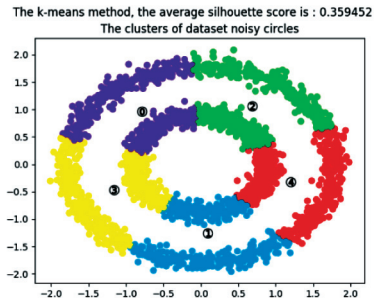
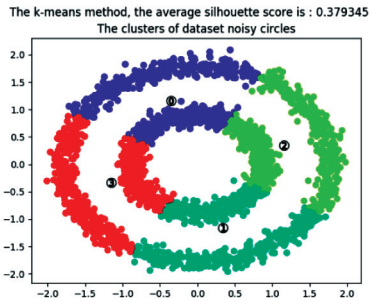
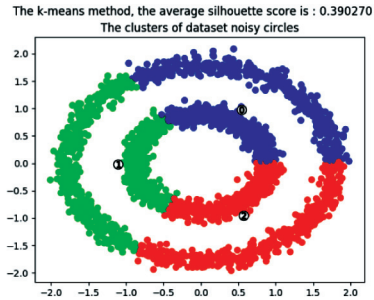
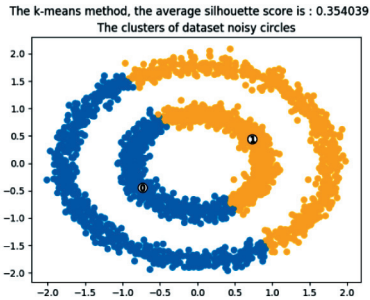
n_samples = 2500
X, y = datasets.make_circles(n_samples=n_samples, factor=.5, noise=.05)
X_stand = StandardScaler().fit_transform(X)

range_n_clusters = [2, 3, 4, 5, 6]
cmap = plt.cm.get_cmap('Dark2')

plt.close('all')
for n_clusters in range_n_clusters:
    fig = plt.figure()
    # k-means method
    clusterer = KMeans(n_clusters=n_clusters, random_state=10)
    cluster_labels = clusterer.fit_predict(X_stand)
    # Compute the average value of silhouette
    silhouette_avg = silhouette_score(X_stand, cluster_labels)
    print("For n_clusters =", n_clusters, "noisy circles",
          "The average silhouette_score is :", silhouette_avg)
    # the Plot showing the actual clusters formed
    colors = cmap((cluster_labels.astype(float) + .5) / n_clusters)
    plt.scatter(X_stand[:, 0], X_stand[:, 1], c=colors)
    plt.title("The clusters of dataset noisy circles")
    # Labeling the clusters
    centers = clusterer.cluster_centers_
    # Draw white circles at cluster centers
    plt.scatter(centers[:, 0], centers[:, 1], marker='o',
               c="white", alpha=1, s=100, edgecolor='k')
    for i, c in enumerate(centers):
        plt.scatter(c[0], c[1], marker='$%d$' % i, alpha=1,
                  s=50, edgecolor='k')
    fig.suptitle("The k-means method, the average silhouette score is : %f "
                 "% (silhouette_avg))

plt.show()

```



#####

# Example 5.15

# spectral clustering

# Generating dataset (noisy circles, Two-dimensional spiral)

#####

import matplotlib.pyplot as plt

import random as rm

import numpy as np

from sklearn.cluster import SpectralClustering

from sklearn.preprocessing import StandardScaler

import warnings

warnings.filterwarnings("ignore")

# three centred noisy circles

X = [(0.5+rm.uniform(-.1,.1))\*np.cos(float(t)/100\*2\*np.pi)+1,(0.5+rm.uniform(-.1,.1))\*np.sin(float(t)/100\*2\*np.pi)] for t in range(0,100)

Y = [(1.5+rm.uniform(-.3,.3))\*np.cos(float(t)/100\*2\*np.pi)+1,(1.5+rm.uniform(-.3,.3))\*np.sin(float(t)/100\*2\*np.pi)] for t in range(0,250)

Z = [(3.5+rm.uniform(-.5,.5))\*np.cos(float(t)/250\*2\*np.pi)+1,(3.5+rm.uniform(-.5,.5))\*np.sin(float(t)/250\*2\*np.pi)] for t in range(0,500)

```
X = X+Y+Z
```

```
X=np.array(X)
```

```
plt.figure(figsize=(12, 12))
```

```
plt.subplot(221)
```

```
plt.scatter(X[:, 0], X[:, 1], s=10)
```

```
colors = np.array([x for x in 'bgrcmykgbgrcmkykgbgrcmkykgbgrcmkyk'])
```

```
colors = np.hstack([colors] * 20)
```

```
spectral = SpectralClustering(n_clusters=3,
```

```
                             eigen_solver='arpack',
```

```
                             affinity="nearest_neighbors")
```

```
X1 = StandardScaler().fit_transform(X)
```

```
spectral.fit(X1)
```

```
y_pred = spectral.fit_predict(X1)
```

```
plt.subplot(222)
```

```
plt.scatter(X[:, 0], X[:, 1], color=colors[y_pred].tolist(), s=10)
```

```
# Two-dimensional spirals
```

```
X = [[(float(t)/200*10+rm.uniform(-.8,.8))*np.cos(float(t)/250*2*np.pi)+1,(float(t)/  
      200*10+rm.uniform(-.8,.8))*np.sin(float(t)/250*2*np.pi)] for t in range(25,500)]
```

```
Y = [[(float(t)/200*10+rm.uniform(-.8,.8))*np.cos(float(t)/250*2*np.pi+np.pi)+1,(float(t)/  
      200*10+rm.uniform(-.8,.8))*np.sin(float(t)/250*2*np.pi+np.pi)] for t in range(25,500)]
```

```
X =X+Y
```

```
X=np.array(X)
```

```
plt.subplot(223)
```

```
plt.scatter(X[:, 0], X[:, 1], s=10)
```

```
spectral = SpectralClustering(n_clusters=2,
```

```
                             eigen_solver='arpack',
```

```
                             affinity="nearest_neighbors")
```

```
X1 = StandardScaler().fit_transform(X)
```

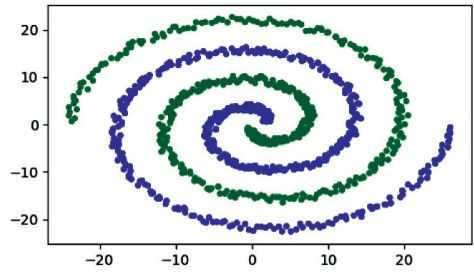
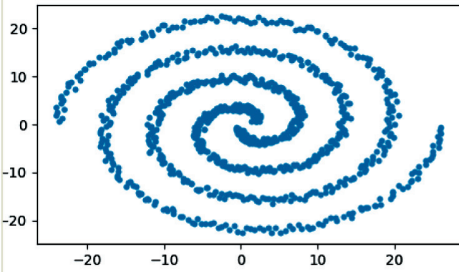
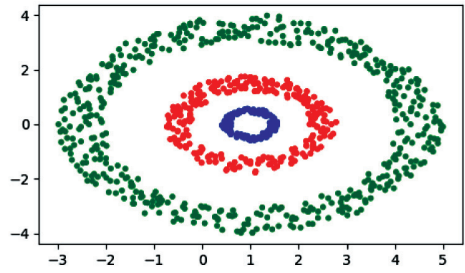
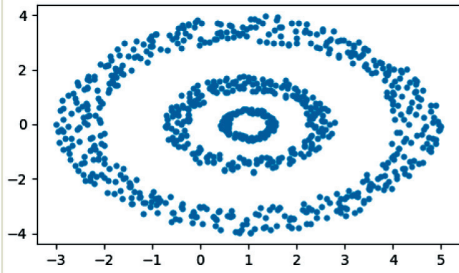
```
spectral.fit(X1)
```

```
y_pred = spectral.fit_predict(X1)
```

```
plt.subplot(224)
```

```
plt.scatter(X[:, 0], X[:, 1], color=colors[y_pred].tolist(), s=10)
```

```
plt.show()
```



## 6. Classifiers

The huge amount of information possessed by mankind resulted in creating the concept of automation of knowledge extraction – Data Mining. This direction is connected with the broad spectrum of tasks starting with recognition of indistinct images before creation of search engines (see in Hastie et al. 2009, Larose 2005, Ponlet et al. 2008, Roiger, Geatz 2008, Theodoridis, Koutroumbas 2006). The important component of Data Mining is processing of text information (see for instance Charu 2015). Such tasks lean on the concept of classification and clustering (see for example Domingos, Pazzani 1997, Ghazanfar, Prügél-Benet 2010, Gordon 1999, Koren et al. 2009, *Naive Bayes...*, Pazzani 1996, Reiten 2017, Rocchio 1966, 1971, Sebastiani 2002). Classification refers to the definition of belonging of some elements to one of the previously created classes. The clustering means splitting the set of elements into clusters which quantity is defined by localization of the set elements in the neighborhood of some natural centers of these clusters. Implementing the classification problem initially has to lean on the set postulates, basic of which are the priori information on primary set of objects and the measure of proximity of elements and classes.

### 6.1. Definition of the classification problem

Let's use the following model of the classification problem, where

$\Omega$  – a set of subjects to recognize (like space of images).

$\omega \in \Omega$  a subject to recognize (for instance: image).

$g(\omega): \Omega \rightarrow \mathfrak{X}$ ,  $\mathfrak{X} = \{1, 2, \dots, n\}$  – the indicator function breaking space of images  $\Omega$  into  $n$  not crossed classes  $\Omega^1, \Omega^2, \dots, \Omega^n$ . Indicator function is unknown to the observer.

$X$  – space of the observations perceived by the observer (space of signs).

$x(\omega): \Omega \rightarrow X$  – the function putting  $\omega$  at the end in compliance to each object  $x(\omega)$  in space of signs. The vector  $x(\omega)$  is the image of the object perceived by the observer.

In space of signs referring to non-overlapping sets of points  $[i] \Xi \subset X$   $i = 1, 2, \dots, n$  are defined groups corresponding to images of one class.

$\varphi(x): X \rightarrow \mathfrak{R}$  – the decisive rule – assessment for  $g(\omega)$  on the basis  $x(\omega)$ , i.e.  $\varphi(x) = \varphi(x(\omega))$ .

Let  $x_\nu = x(\omega_\nu)$ ,  $\nu = 1, 2, \dots, N$  information available to the observer on functions  $g(\omega)$  and  $x(\omega)$ , but these functions are unknown to the observer. Then  $(g_\nu, x_\nu)$ ,  $\nu = 1, 2, \dots, N$  – the set of precedents.

The task involves creating such decisive rule  $\varphi(x)$  whose recognition can be carried out with the minimum number of mistakes.

## 6.2. Main directions of the research of the classification issue

Everyday occurrence is a considering the Euclidean space of signs and quality of the decisive rule to measure, by the frequency of the emergence of the correct decisions. As a rule, the assignment of a set of  $\Omega$  objects to the appropriate class is evaluated with some probability measure. Bayesian approach (see, for example, Domingos, Pazzani 1997, Ghazanfar, Prügel-Benet 2010, John, Langley 1995, Kohavi et al. 1997, Kohavi 1996, Langley et al. 1992, Zhang 2004, Zheng, Webb 2005, Zheng, Webb 2005) proceeds from the statistical nature of observations. The assumption of probability existence measure on space of images which either if is known is taken by the basis, or if is not known can be estimated. The purpose consists of development of such qualifier which can correctly define the most probable class for the trial image. Then the task consists of definition of “the most probable” class. Bayesian approach is based on the assumption of existence of some probability distribution for each parameter. The disadvantage of this method is the need to postulate the existence a priori distribution for an unknown parameter, and its quantitative characteristics.

The use of search of compliance is preceded by creation of the statistician set which contains the number of texts in this class and the list of the used terms together with the counters.

Defining the suitable class of texts for the set text, its structure is under construction of not repeating terms and their cardinalities –  $(w_i, n(w_i))$ .

Let  $M$  denote the size of the number of classes. Let's name the text classes, to which we put phrases, by  $c_j$  ( $j = 0, \dots, M - 1$ ). For each word  $w_i$  from the checked text, we find this word and the corresponding counter in each statistics  $n(w_i, c_j)$  (here  $j = 0, 1, \dots, M - 1$  is the number of the class). Through  $n(c_j)$  let's name the number of texts in  $j$ -th class. Minimization of risk and probability of the mistake are equivalent to division of the space of signs into  $n$  areas. If areas adjacent, then they are divided by the decision surface in multidimensional space. For the case of creating the dividing surface it is more preferable to use discriminatory analyses exactly from Bayesian



hypothesis. The use of probabilistic characteristics is defined on normal distribution which is very widely used because of computing convenience and adequacy in many cases.

If this is known or with a sufficient basis, it can be assumed that the density function determined for the likelihood function  $P(x|\Omega^i)$  is Gaussian, then the use of Bayes classifier leads to the fact that the data characterized by normal distribution show a tendency to group around the average value, and their dispersion is proportional to the mean square deviation  $\sigma$ . Probabilistic methods are guided by information on the elementary probability law for each class. Unfortunately, in real tasks information on density function is absent.

To solve the problem of automatic classification of subjects to space shooting, J. Rocchio (Rocchio 1966, 1971) offered the algorithm TF-IDF (term frequency/inverse document frequency). Let's explain this concept in a few words. TF-IDF is the statistical measure used for assessing the importance of the word in the context of the document which is the part relating to the collection of documents or cases. The weight of some word is proportional to the number of uses of this word in the document and is inversely proportional to the frequency of use of the word in other documents of the collection.

The measure of TF-IDF is often used in tasks of the analysis of texts and the information retrieval, for example, as one of criteria of relevance of the document to the search query, when calculating the measure of proximity of documents at the clustering.

**TF** (*term frequency* – word frequency) is a ratio of the number of occurrences of a word to the total of words in the document. Thus, importance of the word  $t_i$  in the separate document is calculated as follows

$$\text{TF} = \frac{n_i}{\sum_k n_k},$$

where  $n_i$  is the number of the word occurrences in the document, and in the denominator – total number of words in this document.

**IDF** (*inverse document frequency* – the return frequency of the document) – inversion of frequency with which a word occurs in documents of the collection. The accounting of IDF reduces the weight of widespread words

$$\text{IDF} = \log \frac{|D|}{|(d_i \supset t_i)|},$$

where  $|D|$  is the number of documents in the case;  $|(d_i \supset t_i)|$  – the number of documents in which  $t_i$  occurs (when  $n_i \neq 0$ ).

Thus, the measure of TF-IDF is the work of two factors: TF\*IDF. Big weight in TF-IDF will be received by words with high frequency within the specific document and with the low frequency of the uses in other documents.

The first work which laid the foundation for a whole series of the works in this area was the work of Vapnik and Chervonenkis (Vapnik and Chervonenkis 1964, Vapnik and Chervonenkis 1974). The methods of pattern recognition offered by authors and the statistical theory of training, which was their cornerstone were very successful (see more in Vapnik 2000, Sebastiani 2002, Pedregosa et al. 2011, Richert, Coelho 2013, Sammut, Webb 2017). Algorithms of classification and regression under the general name SVM replaced neural networks successfully in many cases and at present are applied very widely.

The idea of the method is based on the assumption that the best way to divide points in  $n$ -dimensional space is  $n - 1$  the plane (set by the  $f(h)$  function), that is equidistant from the points belonging to different classes. The method of basic vectors (Support Vector Machine – SVM) applies to the group of boundary methods. This group of methods defines classes by means of borders of areas. Sets of objects lying on the boundaries of areas are called support vectors. Classification is considered as good if the area between borders is empty. However, the complexity of SVM model creation is that: the higher space dimension is, the more difficult the work with it is. Thus it limits SVM use significantly (see more in section 8).

## 6.3. Stochastic classifiers

### 6.3.1. Use of the theorem of Bayes for decision-making

Let's consider one of the most popular discriminatory analyses based on the stochastic principles. The basis of this method is the priori information on probability distribution of the existing classes  $C_i$ .

The prior (initial) probability is interpreted as the description of information in lack of the certificate of the event. The posterior (subsequent) probability takes this certificate into account. The prior probability of emergence of the event of  $y$  at observation  $x$  is calculated as follows

$$P(y|x) = \frac{P(x|y)P(y)}{P(x)} \quad (6.1)$$

where  $P(x)$  is the prior probability of observation  $x$ , and, respectively  $P(y)$  the prior probability of emergence of the event of  $y$ .

Let's consider the following task: making a decision about gender, i.e. who is a man and who is a woman based on the person's height (see Tab. 6.1).

**Table 6.1**

Data for the decision on the sex i.e. who is a man or who is a woman

Body length	Man	Woman
Dwarfish	it is lower than 129.9 cm	it is lower than 121.9 cm
Very small	130–149.9 cm	121–139.9 cm
Small	150–159.9	140–148.9 cm
Below average	160–163.9 cm	149–152.9 cm
Average	164–166.9 cm	153–155.9 cm
Above average	167–169.9 cm	156–158.9 cm
Big	170–179.9 cm	159–167.9 cm
Very big	180–199.9 cm	168–186.9
Huge	from 200 cm and above	from 187 cm and above

The ethnic origin to certain groups of the people can influence human height. So, for example, the average height of Chinese – 164.8 cm (for men) and 154.5 cm (for women), and the average height of Netherlanders – 184.8 cm and 168.7 cm, respectively. We assumed that the average height of the men on the Earth is 164 cm, and women – 154 cm.

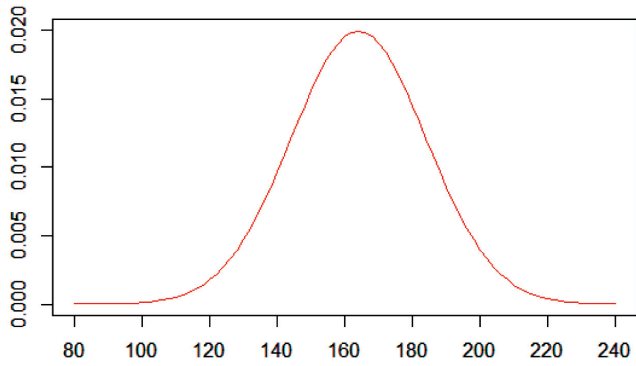
Let's consider that growth of people has normal distribution  $N(\mu, \sigma^2)$ .

$$p(l) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(l-\mu)^2}{2\sigma^2}\right),$$

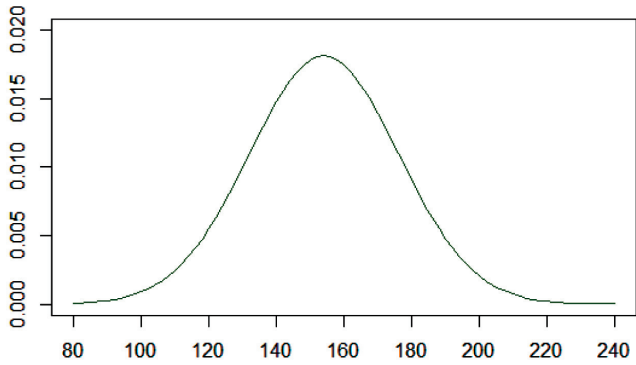
in particular, growth of men has distribution (Fig. 6.1) and for women (Fig. 6.2).

$$p(l | m) = \frac{1}{20\sqrt{2\pi}} \exp\left(-\frac{(l-164)^2}{2(20)^2}\right),$$

$$p(l | w) = \frac{1}{22\sqrt{2\pi}} \exp\left(-\frac{(l-154)^2}{2(22)^2}\right).$$

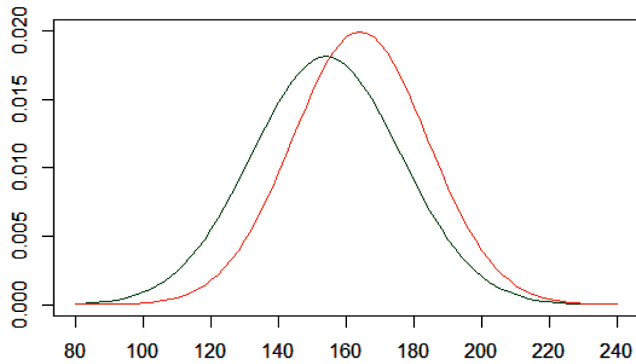


**Fig. 6.1.** The distribution of men height



**Fig. 6.2.** The distribution of women height

It is necessary to evaluate parameters of these distributions in terms of their use in Bayes's theorem decide whether height corresponds to the man or the woman (see Fig. 6.3).



**Fig. 6.3.** The distributions of human height

Let's create a classifier with a maximum probability. It is based on the fact that priority is given to an event for which there is a high probability for a given observation, that is, if an inequality is met for a certain height  $h$

$$p(l | m) > p(l | w),$$

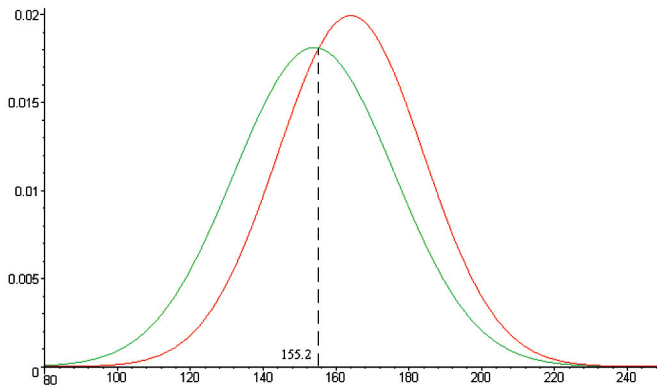
that is

$$\frac{1}{20\sqrt{2\pi}} \exp\left(-\frac{(l-164)^2}{2(20)^2}\right) > \frac{1}{22\sqrt{2\pi}} \exp\left(-\frac{(l-154)^2}{2(22)^2}\right),$$

so we make the decision that we observe the man, otherwise

$$\frac{1}{20\sqrt{2\pi}} \exp\left(-\frac{(l-164)^2}{2(20)^2}\right) < \frac{1}{22\sqrt{2\pi}} \exp\left(-\frac{(l-154)^2}{2(22)^2}\right),$$

we consider that we observe the woman. In this case the equilibrium point in which we cannot define the sex of the person corresponds to the height of 155.2 cm. If the height is less than 155.2 cm, then the qualifier of maximum likelihood considers that this person is a woman if it is more – the male (see Fig. 6.4).



**Fig. 6.4.** The distribution of human height with equilibrium point

The prior probability of the observed height of the man and woman differs. Let's assume that likelihood that a randomly selected person will be a man is  $P(\text{man}) = 2/5$ , and for a woman is  $P(\text{woman}) = 3/5$ .

According to Bayes rule

$$P(m|l) = \frac{p(l|m)P(m)}{p(l)} \text{ and } P(w|l) = \frac{p(l|w)P(w)}{p(l)}.$$

In this case, the qualifier may take the following form if the assumed inequality is met

$$\frac{p(l|m)P(m)}{p(l)} > \frac{p(l|w)P(w)}{p(l)},$$

or that the same condition

$$p(l|m)P(m) > p(l|w)P(w),$$

then the person being verified is the man, in the opposite case is the women.

Thus, solving inequality takes the form as follows

$$\frac{2}{5} \cdot \frac{1}{20\sqrt{2\pi}} \exp\left(-\frac{(l-164)^2}{2(20)^2}\right) > \frac{3}{5} \cdot \frac{1}{22\sqrt{2\pi}} \exp\left(-\frac{(l-154)^2}{2(22)^2}\right),$$

and we receive that if  $l > 173.4$ , we make the decision that the person is a man, if  $l < 173.4$ , the person is a woman.

## 6.4. Naive Bayesian classifier

The Bayesian classifier is based on that the prior probabilities of hypothesis  $P(c_i)$  are known, that is, probability of belonging to a class  $c_i$  ( $i = 1, 2, \dots, k$ ). The Bayesian classifier does not answer the question how to find them (see for instance in Domingos, Pazzani 1997, John, Langley 1995, Kohavi et al. 1997, Langley et. al. 1992, Zheng, Webb 2005, 2008). The naive Bayesian classifier allows to estimate these probabilities, but incorrectly. Sometimes it is even very wrong, but, nevertheless, it is better, than nothing. The most often used area of the naive Bayesian classifier belongs to the problem of text classification where this method allows to receive quite good results (see in *Naive Bayes ...*, Reiten 2017, Sebastiani 2002). The naive Bayesian classifier assumes conditional independence of attributes, in particular, when processing texts, assumptions of the naive qualifier are absolutely discouraging – the probability of emergence

of the word does not depend on other words in the document and, moreover, does not depend on the document size. Surprisingly, the naive Bayesian classifier was quite effective for processing of texts and was widely adopted, in particular, for filtering of spam (for example Ghazanfar and Prügel-Bennet 2010, *Naive Bayes...*, Rocchio 1996, 1971).

Let's pass to the statement and discussion on the naive Bayesian classifier.

It is supposed that the algorithm of classification works on a set of documents  $D = \{b_i\}$ . Each set of documents breaks into disjoint subsets.

$$C = \{c_i\}, \bigcup_i \{b_i\} = D, c_i \cap c_j = \emptyset (i \neq j).$$

Problem of classification is a definition of the class to which a document belongs. For each element  $b$  the feature set is put in compliance with  $\{w_i\}$  which is a set of terms in the document. Further the set of documents defining the class is called the training selection.

Moreover the classification algorithm is applied to allocate documents to the most corresponding class.

Applying Bayes' theorem (6.1) for classification of documents, the following assumptions should be considered

$$P(c_j) = \frac{n(c_j)}{\sum_j n(c_j)},$$

where  $n(c_j)$  is a quantity of terms in the class  $c_j$ .

It is supposed that all terms (words, phrases) are independent, respectively

$$P(w_i | c_j) = \frac{n(w_i, c_j)}{n(c_j)},$$

where:

- $\{w_i\}$  – the set of terms in the document  $b$ ,
- $n(w_i, c_j)$  – quantity of terms  $w_i$  in the class  $c_j$ .

For determining the suitable category of documents for the considered document, it is necessary to receive the corresponding set of words in appropriate form. The structure of the set of word forms includes only non-repetitive words ( $w_i$ ) and their counters ( $n_i$ ).

Determination of suitable category begins with the root of the tree referring to the set of statistics. Let  $M$  denote the quantity of the statistic set in this node of the tree. The categories according to which we check the belonging of documents are marked by  $c_j$  ( $j = 0, \dots, M - 1$ ). For each word  $w_i$  we find this word and the corresponding counter in each set of statistics. Then  $n(w_i, c_j)$  (here  $j = 0, 1, \dots, M - 1$ ) is a number of category (the set of statistics). Through  $n(c_j)$  let's designate number of documents in category  $j$ -th. Besides, let

$$N_j(w_i) = \frac{n(w_i, c_j)}{n(c_j)},$$

be a rated counter of the word  $w_i$  in category  $j$ -th.

Then the probability of matching a unique word  $w_i$  (i.e. each word occurs only once) with  $j$ -th category will be equal

$$P(c_j | w_i) = \frac{N_j(w_i)}{S(w_i)} n(w_i, c_j) \quad (6.2)$$

where

$$S(w_i) = \sum_{j=0}^{M-1} N_j(w_i).$$

If  $P(c_j | w_i) = 0$  then it is necessary to take this number equal to a small value, for example, equal.

Then the probability that the document corresponds to category  $c_j$  ( $j = 0, \dots, M - 1$ ) will be equal

$$P(c_j | \{w_i\}) = P(c_j) \prod_i P(c_j | w_i),$$

where it reflects all words of the studied set of word forms and

$$P(c_j) = \frac{n(c_j)}{\sum_{j=0}^{M-1} n(c_j)}.$$

is the prior probability of meeting category  $c_j$ . Let's notice that if the document contains a large number of words which do not match category  $c_j$  ( $j = 0, \dots, M - 1$ ), that



value  $P(c_j | \{w_i\})$  can go beyond definition of the variable. Therefore, it is necessary to control the value  $P(c_j | \{w_i\})$ , and if it exceeds the value of the variable, then it should be limited to a fixed small number, for example, simply take a value of 0.

If linking words are not discarded (i.e. the ones which do not bear any information on the subject of the text, for example, words “then”, “if”, “but”, etc.), then it makes sense to reduce influence of words which occur in the large number of categories. For this purpose it makes sense to use the following formula

$$P(c_j | w_i) = \frac{N_j(w_i)}{S(w_i)} n(w_i, c_j) \log \frac{M + 1/2}{M(w_i) + 1/2},$$

where  $M(w_i)$  is the amount of categories in which the word  $w_i$  occurs.

Let’s notice that if all categories contain all words of the document, then the algorithm will show discrepancy. Thus, this case (when all categories contain all words) has to be processed with use of the formula (6.2).

After the first step we define  $k$ -pieces of categories with the greatest value  $P(c_j | \{w_i\})$ . We keep their name and values  $P(c_j | \{w_i\})$ . According to the name of each of these sets, we come into the corresponding point and we carry out processing. If information is absent, then this point is missed. After that all categories defined in the first stage will be processed. Then from the newly created and from the categories from the previous stage only  $k$  categories with the highest value  $P(c_j | \{w_i\})$  are selected. We keep their name and values  $P(c_j | \{w_i\})$ , then, we pass to the following step, but only on those categories, which were not kept on the previous step. The process is continued until there are categories on which it is possible to carry out the check.

The result of the program implementation will be selected  $k$  categories, the most probable for the examined document, from the point of view of Bayes criterion.

### 6.4.1. Example of sale of the Naive Bayes classifier

Let’s review the program example of sale employing the naive Bayesian classifier. As a model task<sup>1</sup> we will consider the following one: On many websites devoted to movies, books, goods there is an opportunity to leave responses reviews. Let’s try to teach the classifier to distinguish the negative review from the positive one.

For this purpose we will prepare several examples of positive and negative reviews, i.e., we will make so-called training with the teacher. As parameters by which the

---

<sup>1</sup> The formulation of the task and location of input data it is spotted on the following webpage <http://www.cs.cmu.edu/afs/cs/project/theo-11/www/naive-bayes.html>.

qualifier will try to determine emotional coloring of the text there will be separate words which are present at texts or more precisely, frequencies of their occurrence in the texts.

For simplification of the program, we will make it as a console application, and we will place all examples (training and text) directly in the source code. Let's begin with the general scheme of use of the classifier (see program.py file). In the body of the main class of the program the classifier copy is announced. We will consider it slightly below.

Further there are two examples of negative responses (class “-”) and two positive (class “+”). These examples are transferred to the qualifier for training (creation of statistics). And then the qualifier is tested on one obviously positive and one obviously negative example.

```
from classifier import *
```

```
classifier = Classifier()
```

```
def print_hypothesis(opinion, classifier):
```

```
    print("Review:\n")
```

```
    print(opinion + "\n")
```

```
    for item in classifier.get_hypothesis(opinion):
```

```
        print("Hypothesis: ", item[0], ", Probability: ", item[1], "\n")
```

```
classifier.train("""Actually, there is practically nothing to say about the film,
```

```
    Because in the end we did not even see the movie.
```

```
    Creation of actors alternated with frames in style
```

```
    "How not to shoot a movie," while the scriptwriter
```

```
    Was preparing for the control work on natural sciences,
```

```
    And the highly respected director attended the sessions
```

```
    On correcting the inherent mental inferiority.""", "-")
```

```
classifier.train("""To say that I was satisfied with what I saw would be
```

```
    Not entirely correct, rather the whole story is left
```

```
    Me completely indifferent, which is quite strange.
```

```
    I always watch movies very emotionally. In almost every
```

```
    Film, even the most trashy, there are episodes,
```

```
    Which are remembered to me.""", "-")
```

```
classifier.train("""I went to this film on the very first day of the rental.
```

```
    Expected a lot of just the trailer
```

```
    When I first saw him at the cinema,
```

```

I wanted to see the film itself because of the humor, firstly.
I was not disappointed. Perhaps the only drawback
You can call the story itself – a bit trivial, and cruel.
But everything else, if compared with other
Russian paintings, at a high level. So, I will describe.""", "+")
classifier.train("""Probably, only in mediocre and stereotyped
Things can be seen most important.
Alexander Chernyaev's film "The Irony of Love" – a bright
An example of an eternal pass-through fairytale story,
As the princess falls in love with the usual "Ivan", the plot,
Transferred to our realities. The film can be scolded
As many as you like, but it was once again noticed
Much interesting about the psychology of modern man""", "+")

```

```

opinion = """In general, the film should be perceived as a pleasant
A sweet picture that makes you smile, smile,
Empathize with the heroes. I advise you to watch the girls,
Dreaming of a prince on a white horse and believing in love."""
print_hypothesis(opinion, classifier)

```

```

opinion = """A crappy, boring film. While I watched it, I slept
three times. Do not waste time on this misunderstanding."""
print_hypothesis(opinion, classifier)

```

Let's consider actually algorithmic part of the classifier now – the class Classifier.

The main two methods, which are visible to other classes are the method starting after-training, and the method which is carrying out actually classification namely GetHypothesis(). Let's start with the class fields. Their assignment is included in their names:

```

class Classifier:
    _indices = []
    _classes = []
    _numOfDocs = 0
    _numOfDocsPerClass = []

    _numOfToks = 0
    _numOfToksPerClass = []

```

The method starting training (after-training) is below:

```
def train(self, sample, class_name):

    if class_name in self._classes:
        class_index = self._classes.index(class_name)
    else:
        class_index = len(self._classes)
        self._classes.append(class_name)
        self._numOfDocsPerClass.append(0)
        self._numOfToksPerClass.append(0)
        self._indices.append({})
    self._numOfDocs += 1
    self._numOfDocsPerClass[class_index] += 1
    for token in sample.lower().split():
        self._numOfToks += 1
        if token not in self._indices[class_index]:
            self._indices[class_index][token] = 0
        self._indices[class_index][token] += 1
        self._numOfToksPerClass[class_index] += 1
```

And, at last, the method classifying the text can take the following form:

```
def get_hypothesis(self, opinion):
    result = []
    for class_index in range(0, len(self._classes)):
        prob = self._numOfDocsPerClass[class_index] / self._numOfDocs
        for token in opinion.lower().split():
            if token not in self._indices[class_index]:
                counter = 0
            else:
                counter = self._indices[class_index][token]
            prob *= (counter + 1) / (self._numOfToksPerClass[class_index] + self._numOfToks)
        result.append((self._classes[class_index], prob))
    result.sort(key=lambda element: float(element[1]), reverse=True)
    return result
```

Here we see the algorithm described in the previous chapter. The number of documents in the class during the training was divided by the total number of documents

and assumed as a priori probability. If the a priori probabilities are identical, it is possible to assign in this line just a value 1 (or any other, significantly positive value, different from zero). The relative probabilities of hypotheses, but not their absolute values are important. Calculation of probabilities comes from the number of occurrences “on the fly”. It allows to use easily the class Classifier in the after-training mode when incorrectly recognized text goes to the Train method. Also it is necessary to pay attention to an expression “(counter + 1)”. The added unit allows to avoid loss of accuracy in case of contradictory data when in the evaluated text there are words which are absent in both categories. Without the value of 1, the program would give us zero, and other hits to this class (even in large numbers) would not matter.

Now we start the program and we look at the result:

Review:

In general, the film should be perceived as a pleasant

A sweet picture that makes you smile, smile,  
Empathize with the heroes. I advise you to watch the girls,  
Dreaming of a prince on a white horse and believing in love.

Hypothesis: +, Probability: 1.5660161372743085e-94

Hypothesis: -, Probability: 5.542803672234031e-97

Review:

A crappy, boring film. While I watched it, I slept  
three times. Do not waste time on this misunderstanding.

Hypothesis: -, Probability: 6.468897463772931e-47

Hypothesis: +, Probability: 2.068650180178771e-47

As we see, the result is quite reliable. Of course, you should not expect that this classifier will show big accuracy in all cases. For this purpose it is necessary to make some changes to it. For example, it is necessary to bring words, before their processing, to the main word-form. Or at least to carry out the related lexemes. It is especially urgent for morphologically rich languages e.g. Poland, Russian and Ukrainian.

Further, it is also desirable to consider chains of several words as parameters. In the algorithm, to avoid loss of accuracy at long texts, it is necessary to use adding their logarithms instead of multiplying probabilities (frequencies).

It is possible to offer the set of such improvements, however we will let the readers examine the implementation on their own.

### 6.4.2. EM algorithm

This section have a little difficult history. Most likely, the em-algorithm (EM- expectation–maximization) was explained by A. Dempster, M. Laird and D. Rubin in 1977 (Dempster et al. 1977). These days it is seen as one of the most used methods dedicated for dividing mixed components data (see, for example, Roiger, Geatz 2003, Press et al. 2007, Vapnik 2000, Dempster et al. 1977, Wu 1983, Idris 2014, Pedregosa et al. 2011).

As it was already noted earlier, the Bayesian classifier leans on the fact that the prior probabilities  $p(C_i)$  and conditional probabilities  $p(x|C_i)$  are known. Unfortunately, we do not always have such complete information.

Let’s consider one of the methods to solve this task. On the assumption that only the general view of probability distribution is known, it is necessary to evaluate its parameters. It is more often known that required distribution represents linear combination, for example, of normal or binomial distributions. In this case we say that the distribution represents mixed normal and binomial (or others) distributions. In particular, if it is known that probability density function of each class is a normal distribution  $N(\mu_i, \sigma_i^2)$ , it is necessary to evaluate its parameters  $\mu_i, \sigma_i$ . For the Bayesian classifier these parameters are known.

Let  $p(x|\theta_i)$  be the probability density of the fact that observation is received from  $i$ -th components of the mixture of distributions (see Fig. 6.5).

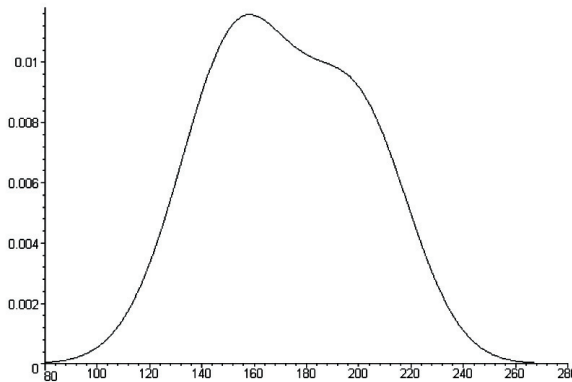


Fig. 6.5. The example of mixed distribution

$$\text{Then } p(x, \theta_i) = p(x)P(\theta_i | x) = \omega_i p_i(x).$$

Posterior probability  $p(\theta_i|x)$  that the observation  $x_j$  is received from  $i$ -th mixed components, we will designate through  $g_{i,j}$ . From the formula of the composite probability we will write out the normalization condition as follows

$$\sum_{i=1}^k g_{i,j} = 1, j = 1, \dots, n.$$

Then, at known  $w_i$  and  $p_i(x_j)$  from Bayes' theorem, it is easy to receive

$$g_{i,j} = \frac{\omega_i p_i(x_j)}{\sum_{v=1}^k \omega_v p_v(x_j)}, i = 1, \dots, k, j = 1, \dots, n.$$

Function

$$F(\Theta) = \prod_{j=1}^n p(x_j, \theta_i),$$

is called the function of credibility from  $\Theta = \{(\omega_i, \theta_i)\}_{i=1}^k$  on selection  $X = \{x_1, \dots, x_n\}$ .

Traditionally using the maximum likelihood method (Maximum Likelihood Estimate – MLE) the model search procedure is called

$$\hat{\Theta} = \arg \max_{\Theta} (F(\Theta)).$$

Let's notice that owing to monotonicity algorithm we get

$$\hat{\Theta} = \arg \max_{\Theta} (F(\Theta)) = \arg \max_{\Theta} (\ln(F(\Theta))).$$

Let's show, that at known  $g_{i,j}$  and using MLE, it is possible to receive the effective method of estimation for the parameters of mixed distribution. Let's write down MLE in the following form

$$\ln F(\Theta) = \ln \prod_{j=1}^n p(x_j, \theta_i) = \sum_{j=1}^n \ln \sum_{i=1}^k \omega_i p_i(x_j) \rightarrow \max_{\Theta}$$

under the condition  $\sum_{i=1}^k \omega_i = 1$ .

We use the method of Lagrange multiplier, i.e. we write out the Lagrangian for this task

$$L(\Theta, \Omega) = \sum_{j=1}^n \ln \sum_{i=1}^k \omega_i p_i(x_j) - \lambda \left( \sum_{i=1}^k \omega_i - 1 \right).$$

Equating partial differential with variables coefficients  $\omega_i$  to zero, we receive

$$\frac{\partial}{\partial \omega_i} L(\Theta, \Omega) = \sum_{j=1}^n \frac{p_i(x_j)}{\sum_{v=1}^k \omega_v p_v(x_j)} - \lambda = 0, \quad i = 1, \dots, k.$$

Multiplying both parts of the received quotient by  $\omega_i$  and summing up them all  $i$ , we get

$$\sum_{j=1}^n \sum_{i=1}^k \frac{\omega_i p_i(x_j)}{\sum_{v=1}^k \omega_v p_v(x_j)} = \lambda \sum_{i=1}^k \omega_i$$

Noticing that

$$\sum_{i=1}^k \frac{\omega_i p_i(x_j)}{\sum_{v=1}^k \omega_v p_v(x_j)} = 1 \quad \text{and} \quad \lambda \sum_{i=1}^k \omega_i = 1,$$

we receive

$$\sum_{j=1}^n 1 = \lambda \quad \text{and} \quad \lambda = n.$$

Thus we get

$$\omega_i = \frac{1}{n} \sum_{j=1}^n \frac{\omega_i p_i(x_j)}{\sum_{v=1}^k \omega_v p_v(x_j)} = \frac{1}{n} \sum_{j=1}^n g_{i,j}, \quad i = 1, \dots, k.$$



Now, noticing that  $p_i(x)$  depends on  $\theta_i$ , we take the partial differential of the Lagrangian where the variables are  $\theta_i$  and we also equate it to zero i.e.

$$\frac{\partial}{\partial \theta_i} L(\Theta, \Omega) = \sum_{j=1}^n \frac{\omega_j}{\sum_{v=1}^k \omega_v p_v(x_j)} \frac{\partial}{\partial \theta_i} p_i(x_j) = 0, \quad i = 1, \dots, k.$$

We will extend each of components of the sum using the logarithmic derivate and the result will be separate by  $p_i(x_j)$  respectively

$$\begin{aligned} \frac{\partial}{\partial \theta_i} L(\Theta, \Omega) &= \sum_{j=1}^n \frac{\omega_j p_i(x_j)}{\sum_{v=1}^k \omega_v p_v(x_j)} \frac{\partial}{\partial \theta_i} \ln p_i(x_j) = \\ &= \sum_{j=1}^n g_{i,j} \frac{\partial}{\partial \theta_i} \ln p_i(x_j) = 0, \quad i = 1, \dots, k. \end{aligned}$$

We receive from here

$$\frac{\partial}{\partial \theta_i} \sum_{j=1}^n g_{i,j} \ln p_i(x_j) = 0, \quad i = 1, \dots, k,$$

which corresponds to the necessary condition of the maximum in the problem of maximizing weighted function of credibility

$$\sum_{j=1}^n g_{i,j} \ln p_i(x_j) \rightarrow \max_{\Theta}, \quad i = 1, \dots, k.$$

Thus the EM (expectation and maximization) algorithm with the fixed number of components of mixed distributions can be written down in the following form.

Let  $X = \{x_1, \dots, x_n\}$  – be a selection of observations,  $k$  – number of components of mixed distributions,  $\Theta = \{(\omega_i, \theta_i)\}_{i=1}^k$  – initial approach of mixed distribution parameters, and  $\varepsilon$  – the number defining the stop algorithm.

The EM algorithm consists of consecutive application of two steps.

### E-step (expectation)

$$g_{i,j}^0 = g_{i,j};$$

$$g_{i,j} = \frac{\omega_i p_i(x_j)}{\sum_{v=1}^k \omega_v p_v(x_j)}, \quad i = 1, \dots, k, \quad j = 1, \dots, n;$$

$$\delta = \max \left\{ \left| g_{i,j}^0 - g_{i,j} \right| \right\}.$$

### M-step (maximization)

$$\sum_{j=1}^n g_{i,j} \ln p_i(x_j) \max_{\Theta}, \quad i = 1, \dots, k;$$

$$\omega_i = \frac{1}{n} \sum_{j=1}^n g_{i,j}, \quad i = 1, \dots, k.$$

If  $\delta > \varepsilon$ , we pass to the E-step, if  $\delta \leq \varepsilon$  we return the found mixed distribution parameters  $\Theta = \{(\omega_i, \theta_i)\}_{i=1}^k$ .

Let's notice that if the type of density function is known, then the task of MLE can be written out in an explicit form. Let's consider the case when it is known that mixed distributions consists of normal distributions  $N(\mu_i, \sigma_i^2)$ , where  $i = 1, \dots, k$ . Then the task

$$\sum_{j=1}^n g_{i,j} \ln p_i(x_j) \rightarrow \max_{\Theta}, \quad i = 1, \dots, k,$$

will be registered in the following form

$$\sum_{j=1}^n g_{i,j} \ln \left( \frac{1}{\sigma_i \sqrt{2\pi}} \exp \left( -\frac{(x_j - \mu_i)^2}{2\sigma_i^2} \right) \right) \rightarrow \max_{\Theta}, \quad i = 1, \dots, k,$$

or that the same

$$G(\{\mu_i, \sigma_i\}_{i=1}^k) = \sum_{j=1}^n g_{i,j} \left( -\ln(\sigma_i \sqrt{2\pi}) - \frac{(x_j - \mu_i)^2}{2\sigma_i^2} \right) \rightarrow \max_{\Theta}, \quad i = 1, \dots, k.$$

Equating the partial differential equations to zero, we receive

$$\frac{\partial}{\partial \mu_i} G(\{\mu_v, \sigma_v\}_{v=1}^k) = -\sum_{j=1}^n g_{i,j} \frac{(x_j - \mu_i)}{\sigma_i^2} = 0,$$

from here

$$\mu_i = \frac{\sum_{j=1}^n g_{i,j} x_j}{\sum_{j=1}^n g_{i,j}}.$$

Similarly

$$\frac{\partial}{\partial \sigma_i} G(\{\mu_v, \sigma_v\}_{v=1}^k) = -\sum_{j=1}^n g_{i,j} \left( \frac{1}{\sigma_i} - \frac{(x_j - \mu_i)^2}{\sigma_i^3} \right) = -\sum_{j=1}^n g_{i,j} \left( \frac{\sigma_i^2 - (x_j - \mu_i)^2}{\sigma_i^3} \right) = 0,$$

thus, from here we receive

$$\sigma_i^2 = \frac{\sum_{j=1}^n g_{i,j} (x_j - \mu_i)^2}{\sum_{j=1}^n g_{i,j}},$$

which allows to receive distribution parameters in an explicit form.

Let's notice that for the multidimensional case normal distribution is described by the following expression

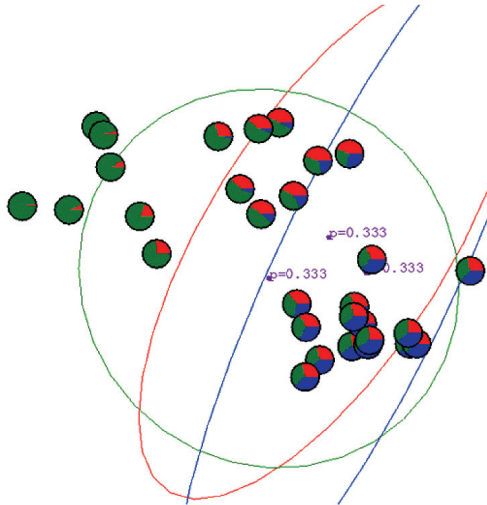
$$p(x | \mu_i, \Sigma_i) = \frac{1}{(2\pi)^{n/2} |\Sigma_i^{-1}|^{1/2}} \exp \left( -\frac{1}{2} (x - \mu_i)^T \Sigma_i^{-1} (x - \mu_i) \right),$$

where  $\Sigma$  is a complete correlation matrix. The use MLE allows to evaluate distribution parameters as follows:

$$\mu_i = \frac{\sum_{j=1}^n g_{i,j} x_j}{\sum_{j=1}^n g_{i,j}},$$

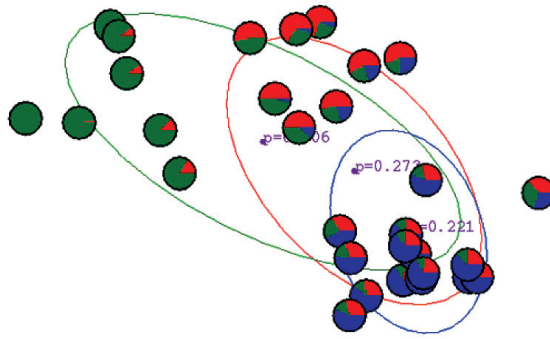
$$\Sigma_i = \frac{\sum_{j=1}^n g_{i,j} (x_j - \mu_i)(x_j - \mu_i)^T}{\sum_{j=1}^n g_{i,j}}.$$

As an illustration of the algorithm “EM” – we show the example given by O. Veksler (Veksler 2006). Suppose that we have 3 – classes, each class is 2-dimensional Gaussian distribution with the same correlation matrix. We start with points shown in Figure 6.6.



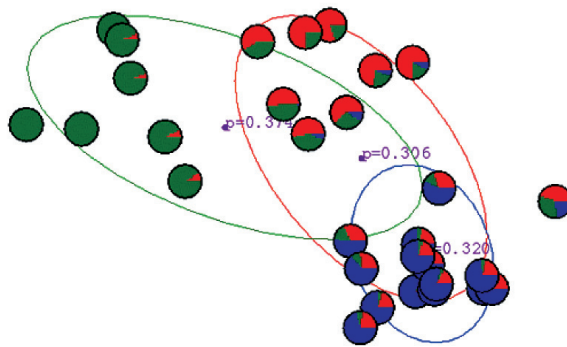
**Fig. 6.6.** Initial approach

We go through the E-step and the M-step and get the adjustment of the distributions shown in Figure 6.7.

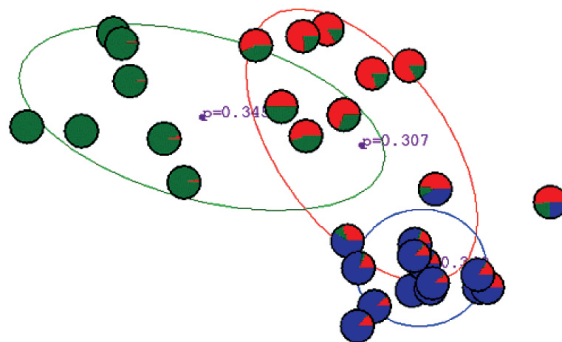


**Fig. 6.7.** The result of the first iteration

In the next iterations the adjustment of the distributions is getting better and better as shown in Figures 6.8–6.10.



**Fig. 6.8.** The result of the second iteration



**Fig. 6.9.** The result after the third iteration

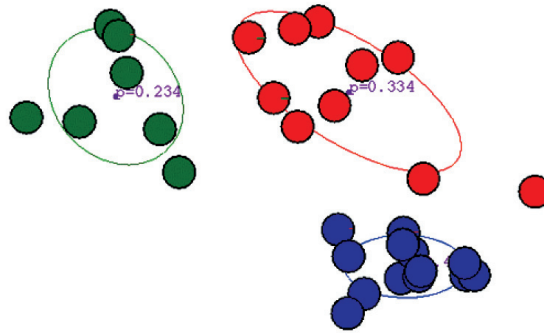


Fig. 6.10. The result of application of twenty iterations

## 6.5. Linear discriminant analysis

Now, when we are able to evaluate parameters of mixed normal distributions, we will consider the problem of classification.

Let  $c_i$  ( $i = 1, 2, \dots, k$ ) be given classes and  $g_i(x)$  a function such that if  $g_i(x) > g_j(x) \forall i \neq j$  then  $x \in c_i$ .

Such function is called discriminant function or function dividing classes.

As discriminant function it is quite naturally used as an initial probability of hit to the class  $c_i$  in the occurrence of the event  $x$

$$g_i(x) = P(c_i | x).$$

Then according to Bayes' theorem (6.1)

$$g_i(x) = \frac{P(x | c_i)P(c_i)}{P(x)}.$$

Because  $P(x)$  does not depend on classes, a discriminant function takes the following form

$$g_i(x) = P(x | c_i)P(c_i).$$

Due to the fact that the logarithm is a monotone function it is possible to use equivalently a discriminant function

$$g_i(x) = \ln P(x | c_i) + \ln P(c_i).$$

If data of the class  $c_i$  are distributed under the normal law  $N(\mu_i, \Sigma_i)$

$$p(x | c_i) = \frac{1}{(2\pi)^{n/2} |\Sigma_i^{-1}|^{1/2}} \exp\left(-\frac{1}{2}(x - \mu_i)^T \Sigma_i^{-1} (x - \mu_i)\right),$$

then

$$g_i(x) = -\frac{1}{2}(x - \mu_i)^T \Sigma_i^{-1} (x - \mu_i) - \frac{n}{2} \ln(2\pi) - \frac{1}{2} \ln |\Sigma_i^{-1}| + \ln P(c_i).$$

As  $n/2 \ln(2\pi)$  is constant, the discriminant function can be written down in the equivalent form

$$g_i(x) = -\frac{1}{2}(x - \mu_i)^T \Sigma_i^{-1} (x - \mu_i) - \frac{1}{2} \ln |\Sigma_i^{-1}| + \ln P(c_i) \quad (6.3)$$

From here, simplifying the expression, we get

$$\begin{aligned} g_i(x) &= -\frac{1}{2}(x^T \Sigma_i^{-1} x - 2\mu_i^T \Sigma_i^{-1} x + \mu_i^T \Sigma_i^{-1} \mu_i) - \frac{1}{2} \ln |\Sigma_i^{-1}| + \ln P(c_i) = \\ &= x^T \left(-\frac{1}{2} \Sigma_i^{-1}\right) x + \mu_i^T \Sigma_i^{-1} x + \left(-\frac{1}{2} \mu_i^T \Sigma_i^{-1} \mu_i - \frac{1}{2} \ln |\Sigma_i^{-1}| + \ln P(c_i)\right) \end{aligned} \quad (6.4)$$

Noticing that

$$x^T Wx = \sum_{i=1}^n \sum_{j=1}^n w_{i,j} x_i x_j,$$

we receive the square function

$$g_i(x) = x^T A_i x + B_i x + D_i,$$

where

$$A_i = -\frac{1}{2} \Sigma_i^{-1}, \quad B_i = \mu_i^T \Sigma_i^{-1}, \quad D_i = -\frac{1}{2} \mu_i^T \Sigma_i^{-1} \mu_i - \frac{1}{2} \ln |\Sigma_i^{-1}| + \ln P(c_i).$$

Thus, the discriminant function will be comprised of the arches of the second order curves (ellipses, parabolas and so forth).

Let's give several special cases of creating the discriminant function.

Let

$$\Sigma_i = \sigma^2 I = \begin{pmatrix} \sigma^2 & 0 & \cdots & 0 \\ 0 & \sigma^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma^2 \end{pmatrix} \quad (i = 1, \dots, k),$$

be random variables ( $X_1, X_2, \dots, X_n$ ) which are independent with the different ensemble average, but with the same dispersion. In this case

$$\Sigma_i^{-1} = \frac{1}{\sigma^2} I = \begin{pmatrix} \frac{1}{\sigma^2} & 0 & \cdots & 0 \\ 0 & \frac{1}{\sigma^2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{1}{\sigma^2} \end{pmatrix} \quad \text{and } |\Sigma_i| = \sigma^{2n}.$$

Noticing that at the same time  $1/2 \ln |\Sigma_i^{-1}|$  is constant, from (6.3) we get the following form of the discriminant function

$$\begin{aligned} g_i(x) &= -\frac{1}{2\sigma^2} (x - \mu_i)^T (x - \mu_i) + \ln P(c_i) = \\ &= -\frac{1}{2\sigma^2} (x^T x - \mu_i^T x - x^T \mu_i + \mu_i^T \mu_i) + \ln P(c_i), \end{aligned}$$

and as  $x^T x$  does not depend on classes, we receive

$$g_i(x) = -\frac{1}{2\sigma^2} (-2\mu_i^T x + \mu_i^T \mu_i) + \ln P(c_i) = a_i x + b_i \quad (6.5)$$

where

$$a_i = \frac{\mu_i^T}{\sigma^2} \quad \text{and} \quad b_i = \ln P(c_i) - \frac{\mu_i^T \mu_i}{2\sigma^2}.$$



Thus, in this case discriminant function is linear, that is, for two variables it is the straight line, for three – the plane, and generally – the hyperplane.

Let's notice that if at the same time all  $P(c_i) = 1/k$ ,  $i = 1, 2, \dots, k$ , symmetric discriminant functions lead to Veronoi splitting.

Furthermore let  $\Sigma_i = \Sigma$  ( $i = 1, 2, 3, \dots, k$ ), then the value  $1/2 \ln |\Sigma_i^{-1}|$  does not depend on the class, and, from (6.3) we receive the following discriminant function

$$\begin{aligned} g_i(x) &= -\frac{1}{2\sigma^2} (x - \mu_i)^T (x - \mu_i) + \ln P(c_i) = \\ &= -\frac{1}{2\sigma^2} (x^T x - \mu_i^T x - x^T \mu_i + \mu_i^T \mu_i) + \ln P(c_i), \end{aligned}$$

As  $x^T \Sigma^{-1} x$  does not depend on classes, we finally receive

$$g_i(x) = -\frac{1}{2} (-2\mu_i^T \Sigma^{-1} x + \mu_i^T \Sigma^{-1} \mu_i) + \ln P(c_i) = a_i x + b_i \quad (6.6)$$

where

$$a_i = \mu_i^T \Sigma^{-1} \quad \text{and} \quad b_i = \ln P(c_i) - \frac{\mu_i^T \Sigma^{-1} \mu_i}{2}.$$

Therefore, in this case the discriminant function is linear too.

### 6.5.1. Example 5.1

Let three classes be described by normal distribution with parameters shown below:

$$\mu_1 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \mu_2 = \begin{pmatrix} 2 \\ 2 \end{pmatrix}, \mu_3 = \begin{pmatrix} 2 \\ -1 \end{pmatrix} \quad \text{and} \quad \Sigma_i = \begin{pmatrix} 4 & 0 \\ 0 & 4 \end{pmatrix}, \quad (i = 1, 2, 3).$$

Besides, let the prior probabilities of loss of classes are given

$$P(c_1) = \frac{5}{12}, \quad P(c_2) = \frac{1}{4}, \quad P(c_3) = \frac{1}{3}.$$

According to (6.5), we will write down the discriminant functions:

$$g_1(x) = \frac{(0,0)}{4} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \left( \ln \frac{5}{12} - \frac{0}{8} \right) = -0.8754683,$$

$$g_2(x) = \frac{(2,2)}{4} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \left( \ln \frac{1}{4} - \frac{8}{8} \right) = -2.3862943 + \frac{1}{2}(x_1 + x_2),$$

$$g_3(x) = \frac{(2,-1)}{4} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \left( \ln \frac{1}{3} - \frac{5}{8} \right) = -1.7261229 + \frac{1}{4}(2x_1 - x_2).$$

Then the function dividing the classes  $c_1$  and  $c_2$  will be equal

$$\begin{aligned} g_1(x) = g_2(x) &\Rightarrow -0.6931471806 = \\ &= -2.098612289 + \frac{1}{2}(x_1 + x_2) \Rightarrow x_1 + x_2 - 3.0217 = 0. \end{aligned}$$

The function dividing classes  $c_1$  and  $c_3$  will be in the following form

$$\begin{aligned} g_1(x) = g_3(x) &\Rightarrow -0.6931471806 = \\ &= -2.416759469 + \frac{1}{4}(2x_1 - x_2) \Rightarrow 2x_1 - x_2 - 3.39257 = 0. \end{aligned}$$

And, at last

$$\begin{aligned} g_2(x) = g_3(x) &\Rightarrow -2.098612289 + \frac{1}{2}(x_1 + x_2) = \\ &= -2.416759469 + \frac{1}{4}(2x_1 - x_2) \Rightarrow x_2 = 0.883576. \end{aligned}$$

Let's notice that  $g_1(x) = g_2(x) = g_3(x)$  has the solution of  $x_1 = 2.138075$ ,  $x_2 = 0.883576$ . It is illustrated in the Figure 6.11.

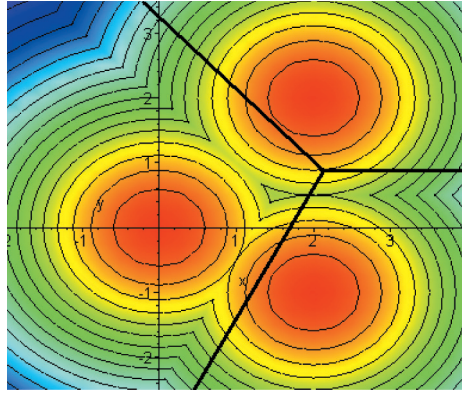


Fig. 6.11. The first example of symmetric discriminant functions

### 6.5.2. Example 5.2

Let three classes be described by normal distribution with parameters put below

$$\mu_1 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \mu_2 = \begin{pmatrix} 2 \\ 2 \end{pmatrix}, \mu_3 = \begin{pmatrix} 2 \\ -1 \end{pmatrix} \text{ and } \Sigma_i = \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix}, (i = 1, 2, 3),$$

furthermore let the prior probabilities of loss of classes take the following values

$$P(c_1) = \frac{5}{12}, P(c_2) = \frac{1}{4}, P(c_3) = \frac{1}{3}.$$

According to (6.4), we will write out discriminant functions as follows:

$$g_1(x) = -0.8754683,$$

$$g_2(x) = -5.3862943 + 2(x_1 + x_2),$$

$$g_3(x) = x - 2.09861229.$$

Then the function dividing classes  $c_1$  and  $c_2$  will be in the following form

$$g_1(x) = g_2(x) \Rightarrow x_1 + x_2 - 2.2554128 = 0.$$

The function dividing classes  $c_1$  and  $c_3$  will be equal

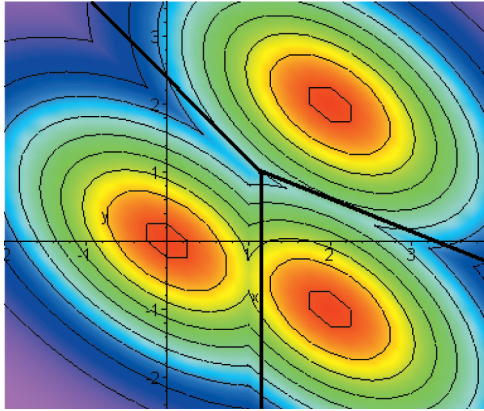
$$g_1(x) = g_3(x) \Rightarrow x_1 - 1.2231435 = 0.$$

And, at last

$$g_2(x) = g_3(x) \Rightarrow x_1 + 2x_2 = 3.28768.$$

In this example  $g_1(x) = g_2(x) = g_3(x)$  has the solution of  $x_1 = 1.22314255$ ,  $x_2 = 1.03226926$ .

Received result is shown in Figure 6.12.



**Fig. 6.12.** The second example of symmetric discriminant functions

### 6.5.3. Example 5.3

Let's take three classes whose data are described by normal distribution with parameters displayed below.

$$\mu_1 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \mu_2 = \begin{pmatrix} 2 \\ 2 \end{pmatrix}, \mu_3 = \begin{pmatrix} 2 \\ -1 \end{pmatrix} \text{ and } \Sigma_1 = \begin{pmatrix} 2 & -1 \\ -1 & 1 \end{pmatrix}, \Sigma_2 = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}, \Sigma_3 = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix},$$

and a priori probabilities of loss of classes which have the following value

$$P(c_1) = \frac{5}{12}, P(c_2) = \frac{1}{4}, P(c_3) = \frac{1}{3}.$$

According to (6.4), we will write down discriminant functions:

$$g_1(x) = -\frac{1}{2}x_1^2 - x_1x_2 - x_2^2 - 0.875468,$$

$$g_2(x) = -\frac{1}{4}x_1^2 - \frac{1}{4}x_2^2 + x_1 + x_2 - 2.693147.$$

Then the function dividing classes  $c_1$  and  $c_2$  can be equal

$$g_1(x) = g_2(x) \Rightarrow -\frac{1}{4}x_1^2 - x_1x_2 - \frac{3}{4}x_2^2 - x_1 - x_2 + 1.81767 = 0.$$

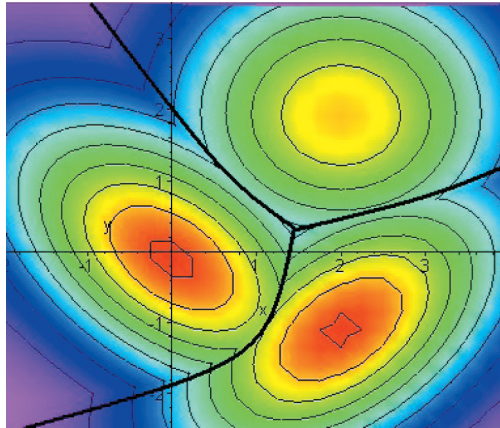
The function dividing classes  $c_1$  and  $c_3$  can take the following form

$$g_1(x) = g_3(x) \Rightarrow -\frac{1}{6}x_1^2 - \frac{4}{3}x_1x_2 - \frac{2}{3}x_2^2 - \frac{5}{3}x_1 + \frac{4}{3}x_2 + 2.00717 = 0.$$

And, at last

$$g_2(x) = g_3(x) \Rightarrow \frac{1}{12}x_1^2 + \frac{1}{12}x_2^2 - \frac{1}{3}x_1x_2 + \frac{7}{3}x_2 - \frac{1}{6}x_1 + 0.18949 = 0.$$

Received result is shown in Figure 6.13.



**Fig. 6.13.** The example of square discriminant functions

## 7. Use of genetic algorithms for creation of the vector classifiers

In this paragraph we will consider the discriminatory analysis based not on the concept of distance between elements but on the criterion of proximity constructed on calculating a cosine of the angle between two vectors. On the example of the classification of texts will be given a detailed algorithm. (see in Slaton, Buckley 1988, Shumeyko, Sotnik 2009, 2011, Swidan et al. 2013 and also in Charu 2015, Layton 2017, Raschka 2015).

The first step consists of preprocessing the data – creating sets of statistics for the available classes. For the creation of the set of statistics all sets of word forms  $b^v$ ,  $v = 0, \dots, M - 1$  are processed consistently, belonging to one class  $B = \{b^v\}_{v=0}^{M-1}$ . On the set of word forms of each processed text  $b^v$  a set of unique (not repeating) word forms and their counters is built  $(w_i^v, n_i^v)(i = 0, \dots, N^v - 1)$ , where  $N^v$  – number of unique word forms in the text  $b^v$ . After that data for each document are normalized separately in the following form

$$\bar{n}_i^v = \frac{n_i^v}{\sqrt{\sum_{j=0}^{N^v-1} (n_j^v)^2}} \quad (i = 0, \dots, N^v - 1).$$

Then, we arrange all words of each document in the same order (the word order is not essential, the main thing is that words in each of structures  $(w_i^v, n_i^v)(i = 0, \dots, N^v - 1)$  have to be the same order) and we find the sum of all vectors  $n_i(B) = \sum_{j=0}^{M-1} \bar{n}_i^v$  ( $i = 0, \dots, N(B)$ ) (where  $N(B)$  – a quantity of unique word forms in class B in general) also we normalize it by its unit as follows

$$\bar{n}_i(B) = \frac{n_i(B)}{\sqrt{\sum_{j=0}^{N(B)} (n_j(B))^2}}.$$

For the received central point of the class we create the set of statistics, writing down in it values  $(w_i(B), \bar{n}_i(B))(i = 0, \dots, N(B))$ .

For creating the central vector of classes  $\{B^\mu\}_{\mu=0}^{K-1}$  where each class  $B^\mu$  is described by the central vector  $(w_i(B^\mu), \bar{n}_i(B^\mu))(i = 0, \dots, N(B^\mu))$ , we need to find their sum. Having summed up all coordinates from all vectors for each value of the word form, for the word form  $\omega$  we receive the coordinate

$$n(\omega) = \sum_{\mu=0}^{K-1} \left\{ \bar{n}_i(B^\mu) \mid \omega_i(B^\mu) = \omega, \quad i = 0, \dots, N(B^\mu) \right\}.$$

Therefore, it is necessary to make the list of unique word forms on all central vectors of classes  $\{B^\mu\}_{\mu=0}^{K-1}$  and to sum up their coordinates. The set consisting their unique (not repeating) word forms and their coordinates can be result

$$\left( w_i \left( \left\{ B^\mu \right\}_{\mu=0}^{K-1} \right), n_i \left( \left\{ B^\mu \right\}_{\mu=0}^{K-1} \right) \right) \left( i = 0, \dots, N \left( \left\{ B^\mu \right\}_{\mu=0}^{K-1} \right) \right),$$

where  $N \left( \left\{ B^\mu \right\}_{\mu=0}^{K-1} \right)$  is a quantity of unique word forms of the set of classes  $\{B^\mu\}_{\mu=0}^{K-1}$ . It is necessary to normalize the received coordinates

$$\bar{n}_i \left( \left\{ B^\mu \right\}_{\mu=0}^{K-1} \right) = \frac{n_i \left( \left\{ B^\mu \right\}_{\mu=0}^{K-1} \right)}{\sqrt{N \left( \left\{ B^\mu \right\}_{\mu=0}^{K-1} \right) \left( \sum_{j=0}^{K-1} \left( n_j \left( \left\{ B^\mu \right\}_{\mu=0}^{K-1} \right) \right)^2 \right)}},$$

and, the received vector  $\left( w_i \left( \left\{ B^\mu \right\}_{\mu=0}^{K-1} \right), \hat{n}_i \left( \left\{ B^\mu \right\}_{\mu=0}^{K-1} \right) \right) \left( i = 0, \dots, N \left( \left\{ B^\mu \right\}_{\mu=0}^{K-1} \right) \right)$  can be the central vector of the set  $\{B^\mu\}_{\mu=0}^{K-1}$ .

Ideally created classification of the vector method is such set of classes  $\{B^\mu\}_{\mu=0}^{K-1}$ , for which the following condition is satisfied:  $\forall b \in B^\mu, \mu = 0, \dots, K-1$

$$\bar{n}(b), \bar{n}(B^\mu) < \bar{n}(b), \bar{n}(B^\nu), \quad \nu \neq \mu \quad (7.1)$$

Let's consider the vector  $\Lambda$  (control vector) of dimension  $N(B^\mu)$ , which coordinates accept only one of two admissible values

$$\lambda_i = \begin{cases} 0 \\ 1 \end{cases}.$$

Through  $\Lambda b$  let's designate the direct product of vectors  $\Lambda$  and  $b$ , that is

$$\Lambda b = \left( \lambda_0 \bar{n}_0(b), \lambda_1 \bar{n}_1(b), \dots, \lambda_{N(B^\mu)} \bar{n}_{N(B^\mu)}(b) \right).$$

Let's call the control  $\Lambda$  admissible on the class  $B^\mu = \{b^k\}_{k=0}^{M-1}$ , if the condition (7.2) is satisfied

$$\overline{\Lambda n}(b^k), \overline{\Lambda n}(B^\mu) < \overline{\Lambda n}(b^k), \bar{n}(B^\nu), \nu \neq \mu, k = 0, 1, \dots, M-1 \quad (7.2)$$

Admissible control vector  $\Lambda$  which fulfils this inequality (7.2) and at the same time fulfils the condition  $\sum_{k=0}^{M-1} (\Lambda b^k)^2 \rightarrow \max$ , is called the optimum.

If for  $\nu \neq \mu$  the set of admissible controls is degenerated, the class  $B^\mu = \{b^k\}_{k=0}^{M-1}$  is defined incorrectly, i.e. it is inseparable from the class  $B^\nu$ .

The problem of finding the optimum control by classical methods is rather difficult therefore we will apply genetic algorithms to this tasks (see Rana 1999).

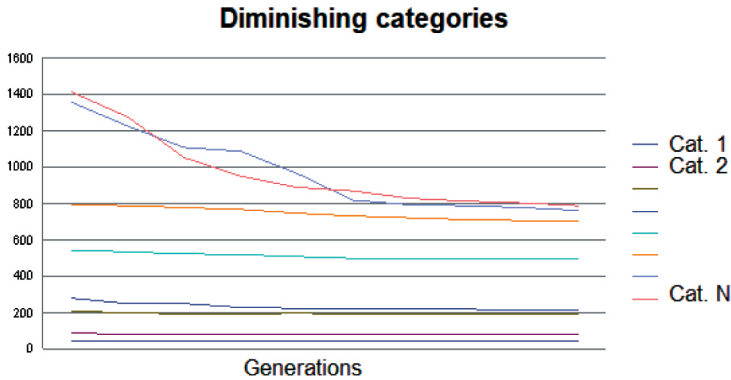
For that matter, the single-point crossing over (single-point crossover) is used. It is modeled as follows: Let there are two parent individuals with chromosomes  $X = \{x_i, i \in \{0, \dots, L\}\}$  and  $Y = \{y_i, i \in \{0, \dots, L\}\}$ . In a random way the point in the chromosome is defined (discontinuity point) in which both chromosomes are divided into two parts and we exchange them. After processing reproduction we can get mutations. It is reached because accidentally chosen gene in the chromosome changes.

For creating a new population we use the elite selection. Intermediate population which includes both parents, and their descendants is created. Members of this population are evaluated, we get out from them the best  $N$  ones (suitable) which will enter the next generation.

The result of applying the genetic algorithm to the problem of reducing class dimension, is given in Figure 7.1.

Let's notice that the vector method as a criterion of quality uses the size of the scalar product of basis vectors, thus, the class of unit vectors (documents) is limited on the sphere by the circle with the center at the end of the central vector of the class.





**Fig. 7.1.** The chart of reduction of dimension of categories when using genetic algorithms

As sphere cuts on the circle cannot densely pack all surface of the single sphere, there is a point set (basis vectors) which cannot essentially get to one class. Thus, there is the need to break the point set on the single sphere so that elements of this splitting can pack all surface of the single sphere densely, which allows to classify any document unambiguously.

## 7.1. Use of Veronoi polyhedron in the problem of texts classification

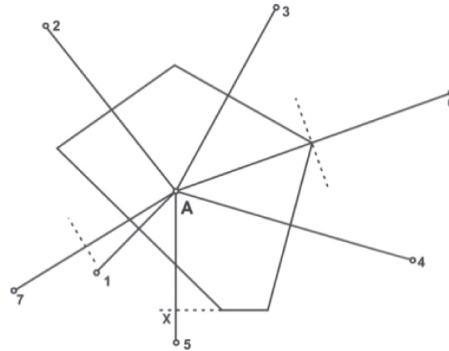
For any center of a system  $\{A\}$  it is possible to specify an area of space where all points are closer to this center, than to any other center of the system. Such area is called the Veronoi polyhedron or Veronoi area. The Veronoi area usually carries to the polyhedron also its outer surface. In three-dimensional space the Veronoi area for any  $i$  center of systems  $\{A\}$  is a convex polyhedron, in two-dimensional space it is a convex polygon. Formally Veronoi polygons  $T_i$  in  $R^2$  are defined as follows

$$T_i = \{x \in R^2 : d(x, x_i) < d(x, x_j) \forall j \neq i\},$$

where  $d$  is a distance function.

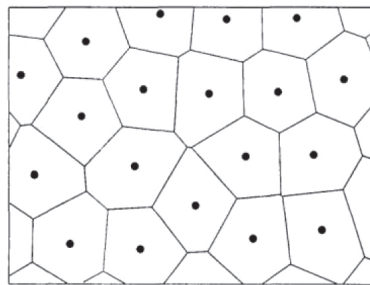
Creating an approximation relies on fundamental property for randomly selected  $n$  set points on the plane  $S$ . For any node from  $n$  on the plane, there is the great number of natural neighbors  $N$ . The concept of natural neighbors is closely connected with splitting the Veronoi area cells. For the nonempty Veronoi cell  $V(R)$ , where  $R \subset S$  the natural neighbors for a vertex of Delon's triangles  $r \in R$ , are the points incidental to  $V(R)$ .

The two-dimensional Veronoi polyhedron (polygon on the plane) is shown in the Figure 7.2. The Veronoi lines which generate edges of the polygon are called the forming lines and the relevant centers of system are the geometrical neighbors of the center  $A$ . Among geometrical neighbors (natural) we can distinguish two kinds of them. For the first kind the bisecting point of a segment connecting it to the central node lies on the verge of the Veronoi polyhedron. For the second kind the bisection point is out of the edge and, therefore, out of the polyhedron.



**Fig 7.2.** The Veronoi polyhedron (polygon) for the center  $A$  in two-dimensional system

Voronoi polyhedrons, the systems constructed for each center  $\{A\}$ , give the mosaic of polyhedrons – splitting Veronoi points (see Fig. 7.3). Veronoi polyhedrons systems  $\{A\}$  do not enter each other and fill the space, being adjacent on the whole edges. Splitting space into Veronoi polyhedrons unambiguously is defined by system  $\{A\}$  and vice versa clearly defines the system.



**Fig. 7.3.** The Veronoi chart on the plane

Using the design of Veronoi charts in relation to points on the multidimensional single sphere, we receive splitting all basis vectors of documents into natural classes.

Borders of the classes will be the hyperplanes dividing spherical Veronoi polyhedrons. At the same time, points on the single sphere (ends of basis vectors of documents) which in relation to all hyperplanes limiting this class lie on the same side of sphere, will belong to one class, as the central vector of this class.

## 7.2. Check of the existing classification on the correctness

Let classes of documents be checked for the splitting correctness  $C_v$  and  $C_\mu$ . For corresponding basis vector (the central vectors)  $\hat{C}_v, \hat{C}_\mu$ , we build a difference vector

$$\bar{\Delta}_{v,\mu} = \hat{C}_v - \hat{C}_\mu = \{ \hat{n}^v(w_i) - \hat{n}^\mu(w_i) \}$$

and a sum vector

$$\bar{\Xi}_{v,\mu} = \frac{1}{2}(\hat{C}_v + \hat{C}_\mu) = \frac{1}{2}\{ \hat{n}^v(w_i) + \hat{n}^\mu(w_i) \}.$$

The new vector is created as the arithmetic mean of two vectors. Let's designate it through  $\bar{\Xi}_{v,\mu}$ . Let's lead a plane through the point  $\bar{\Sigma}_{v,\mu}$  with the normal vector  $\bar{\Delta}_{v,\mu}$ .

$$\Omega_{v,\mu} = \langle \bar{\Delta}_{v,\mu} \cdot (P - \bar{\Xi}_{v,\mu}) \rangle = 0 \quad (7.3)$$

This plane splits the classes. To have classes splitted correctly, all points (documents) of one class should be on the one side of the plane, that is if  $b \in C_v$ , then

$$\langle \bar{\Delta}_{v,\mu} \cdot (\hat{C}_v - \bar{\Xi}_{v,\mu}) \rangle \langle \bar{\Delta}_{v,\mu} \cdot (\hat{b} - \bar{\Xi}_{v,\mu}) \rangle \geq 0.$$

The points for which this condition is not satisfied need to be considered with their belonging to category  $C_\mu$ .

The circumstance when categories have non-empty crossing is possible. For example, the category „AUTHOR” contains documents on mathematics (this author) and the category „SCIENCE” contains documents on mathematics of the same author. In this case it is necessary to allocate non-empty crossing of these categories and, further, either to localize this category, or to address it to both categories.

For solving this problem the following method can be taken into account. Let's consider categories  $C_v$  and  $C_\mu$ . Let's split them by the plane (7.3) collect and all points lying on the one side in two new categories  $C_v^*$  and  $C_\mu^*$ .

Let

$$d(B, \Omega_{v,\mu}) = \frac{\left| \langle \bar{\Delta}_{v,\mu} \cdot (B - \Xi_{v,\mu}) \rangle \right|}{|\bar{\Delta}_{v,\mu}|}$$

be a distance from the point  $B = \{b_i\}$  to the plane  $\Omega_{v,\mu}$ .

If the condition is satisfied (that is, after cutting off of data both categories are removed from each other)

$$\begin{cases} d(C_v^*, \Omega_{v,\mu}) - d(C_v, \Omega_{v,\mu}) > 0, \\ d(C_\mu^*, \Omega_{v,\mu}) - d(C_\mu, \Omega_{v,\mu}) > 0, \end{cases}$$

then categories  $C_v$  and  $C_\mu$  have nonempty crossing  $\tilde{C}$ , which can be defined as follows  $b \in \tilde{C}$  if  $b \in C_v$  and at the same time

$$\langle \bar{\Delta}_{v,\mu} \cdot (\hat{C}_v - \Xi_{v,\mu}) \rangle \langle \bar{\Delta}_{v,\mu} \cdot (\hat{b} - \Xi_{v,\mu}) \rangle < 0,$$

or, if  $b \in C_\mu$  and

$$\langle \bar{\Delta}_{v,\mu} \cdot (\hat{C}_\mu - \Xi_{v,\mu}) \rangle \langle \bar{\Delta}_{v,\mu} \cdot (\hat{b} - \Xi_{v,\mu}) \rangle < 0.$$

It is natural that the problem of classes dimension reduction is also urgent for the method constructed on Veronoi charts. For this purpose, as well as in the previous case, genetic algorithms were used.

Comparative analysis of application of different discriminatory analyses to test base of documents by Reuters (see *Reuters-21578 text categorization...*) is given in the following figures (see Fig. 7.4–7.6).

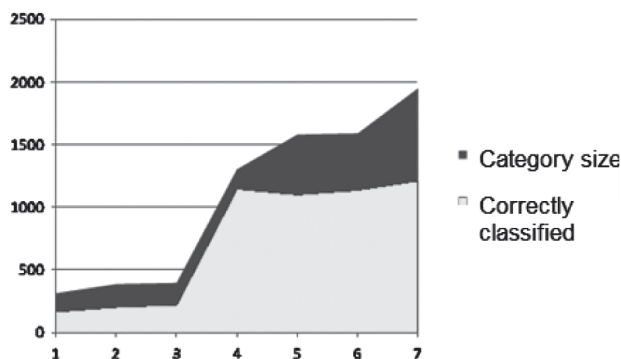


Fig. 7.4. The result of applying the algorithm of Bayes

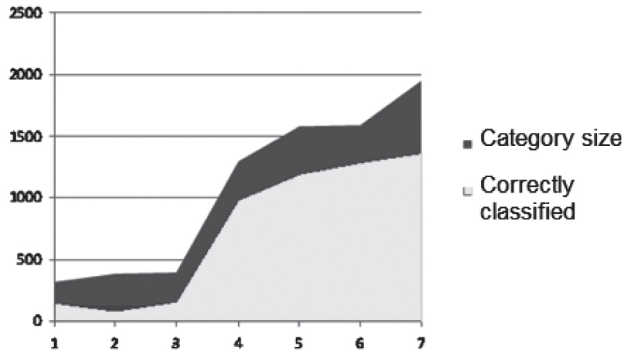


Fig. 7.5. The result of applying the vector algorithm

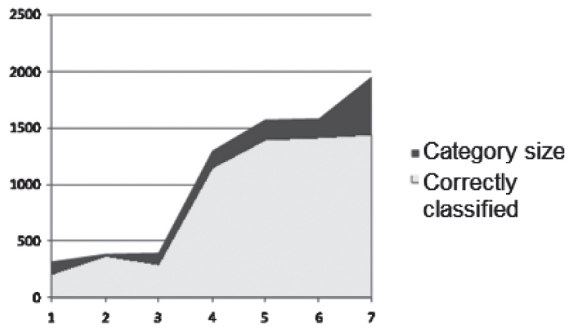


Fig. 7.6. The result of applying the algorithm based on Veronoi charts

So, to test the Reuters base, provided they hit the right class of not less than 90% documents they managed to reduce the class space by about 10% to 50%.

# 8. Support vector machines

The method of support vectors (SVM – Support Vector Machines) is included in the set of algorithms called “supervised learning”. They are effectively used in problems of classification. In 1963 Vapnik and Learner (Vapnik, Learner 1963) proposed an algorithm which finds the maximal margin between the fixed boundary and the nearest points of each class if the data are linearly separable. The next work by Vapnik and Chervonenkis is the cornerstone of SVM. (see Vapnik, Chervonenkis 1964, 1974, Vapnik, Learner 1963, Vapnik 1998, 2000, Chapelle, Vapnik 1999, and also Charu 2015, Han et al. 2011, Hastie 2009, Cristianini, Shawe-Taylor 2000, Joachims 2009, Pedregosa et al. 2011). SVM method belongs to the family of linear qualifiers (*Support Vector Machine...*).

## 8.1. The main idea

In the previous section (see section 2 or section 6) we considered the simplest discriminant functions realizing the linear qualifier (see Fig. 8.1). It can be written down in the form  $g(x) = w^t x + w_0$ , where

$$g(x) > 0 \Rightarrow x \in \text{Class}[1] \text{ and } g(x) < 0 \Rightarrow x \in \text{Class}[2].$$

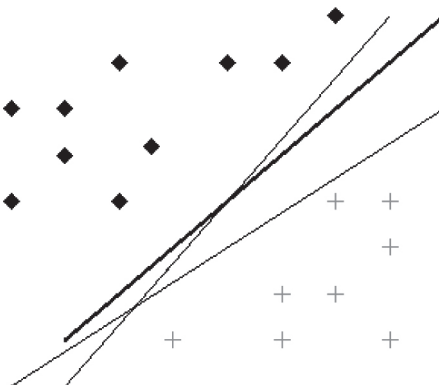


Fig. 8.1. Symmetric determinant functions

Thus the dividing (discriminant) function is described by the equation  $g(x)=0$ . The distance between the point  $x$  and the point of dividing function  $g(x)=0$  is equal

$$\frac{|w^t x + w_0|}{\|w\|}.$$

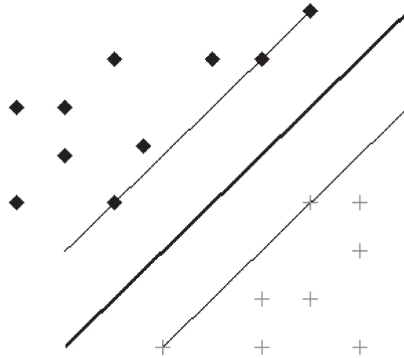


Fig. 8.2. The maximum dividing corridor

Let  $x_i$  lie on short circuit of border, that is  $|w^t x_i + w_0| = 1$ . Border width of the dividing margin, is chosen as wide as possible (see Fig. 8.2). Considering that short strip of border meets the condition  $|w^t x_i + w_0| = 1$ , the distance from  $x_i$  to  $g(x)=0$  is

$$\frac{|w^t x + w_0|}{\|w\|} = \frac{1}{\|w\|} \tag{8.1}$$

thus, width of the dividing strip is equal  $2/\|w\|$  (see Fig. 8.3).

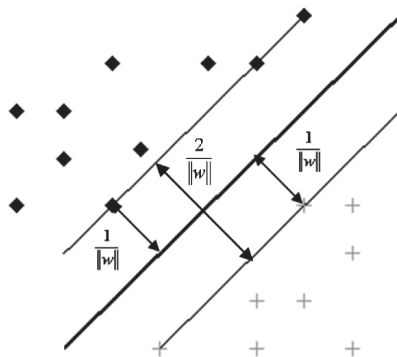


Fig. 8.3. Illustration of creating support vectors

To exclude points from the dividing margin, we will write out the condition of belonging to classes:

$$\begin{cases} w^t x_i + w_0 \geq 1 & \text{if } x_i \text{ belongs to class 1,} \\ w^t x_i + w_0 \leq -1 & \text{if } x_i \text{ belongs to class 2.} \end{cases}$$

Let's enter the index function:

$$\begin{cases} u_i = 1 & \text{if } x_i \text{ belongs to class 1,} \\ u_i = -1 & \text{if } x_i \text{ belongs to class 2.} \end{cases}$$

Thus, the problem of estimating the dividing function generating the corridor of the greatest width is can be written down as a problem of minimizing in the following form

$$J(w) = \frac{1}{2} \|w\|^2 \rightarrow \min \tag{8.2}$$

under the condition  $u_i (w^t x_i + w_0) \geq 1$  for all  $i$ .

The objective function is a square function, so this task has the only one resolve.

## 8.2. SVM for linear separable set

According to Kuhn-Tucker's theorem the condition (8.2) is equivalent to the following

$$L(\alpha) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j u_i u_j x_i^T x_j \rightarrow \max \tag{8.3}$$

provided that  $\alpha \geq 0$  for all  $i$  and  $\sum_{i=1}^n \alpha_i u_i = 0$ , where  $\alpha = \{\alpha_1, \dots, \alpha_n\}$  are new variables. Let's rewrite  $L(\alpha)$  in the matrix form

$$L(\alpha) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix}^T H \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix},$$

where coefficients of the matrix H are calculated as follows

$$H_{i,j} = u_i u_j x_i^T x_j.$$

The task  $L(\alpha) \rightarrow \max$  is solved by methods of quadratic programming.



After finding the optimum  $\alpha = \{\alpha_1, \dots, \alpha_n\}$  for each  $i$  the conditions are verified:

- $\alpha_i = 0$  (it corresponds  $i$  that is not a support vector),
- $\alpha_i \neq 0$  and  $u_i (w^t x_i + w_0 - 1) = 0$  (it corresponds  $i$  that is a support vector).

Then  $w$  from the ratio 8.1 can be found  $w = \sum_{i=1}^n \alpha_i u_i x_i$  and the value  $w_0$  is calculated considering that for any  $\alpha_i > 0$  and  $\alpha_i [u_i (w^t x_i + w_0) - 1] = 0$

$$w_0 = \frac{1}{u_i} - w^t x_i.$$

Then, at last, the discriminant function is received

$$g(x) = \left( \sum \{ \alpha_i u_i x_i \mid x_i \in S \} \right)^T x + w_0.$$

It is important to notice that summing is carried out not on all vectors but only on the set  $S$  which represents the set of support vectors i.e.  $S = \{x_i \mid \alpha_i \neq 0\}$ .

### 8.3. SVM for nonlinear separable set

Unfortunately, the described above algorithm is implementable only for linearly separable sets. In practice such sets are not met frequently. In 1995 Cortes and Vapnik proposed modified algorithm for solving the problem for nonlinear separable sets (see Cortes, Vapnik 1995, Boser et al. 1992, Chapelle, Vapnik 1997, Hamel 2009). Let's look at the modernization of the algorithm for the case of nonlinearly separable sets.

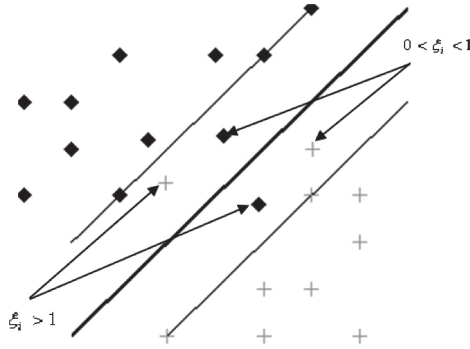
In order to allow misclassification in the model, additional variables  $\xi_i$ , which characterize the mistake size on each object of  $x_i$  are entered. In the objective function, the penalty for the aggregate error is introduced by the following form

$$\begin{cases} \frac{1}{2} \|w\|^2 + \lambda \sum_{i=1}^n \xi_i \rightarrow \min \\ u_i (w^t x_i + w_0) \geq 1 - \xi_i, \quad i = 1, \dots, n \\ \xi_i \geq 0, \quad i = 1, \dots, n \end{cases} \quad (8.4)$$

where  $\lambda$  is the parameter specifying the cost of misclassifications. It allows to control the relation between maximizing width of the dividing strip and minimization of the aggregate error (see Boser et al. 1992, Hamel 2009).

Penalty size  $\xi_i$  for the corresponding object  $x_i$  depends on the arrangement of the object in dividing strip. So, if  $x_i$  lies on the opposite side of discriminant function, then

the penalty size is  $\xi_i > 1$ . If  $x_i$  lies in the dividing strip, but on the same side of discriminant function as the class, then the corresponding weight can take a value  $0 < \xi_i < 1$ . For the ideal separable case the penalty size is  $\xi_i < 0$  (see Fig. 8.4).



**Fig. 8.4.** Points to which penalties are applied

Then the task (8.4) can be rewritten in the form below (see Boser et al. 1992)

$$J(w, \xi_1, \dots, \xi_n) = \frac{1}{2} w^2 + \beta \sum_{i=1}^n I(\xi_i > 0) \rightarrow \min \quad (8.5)$$

that is in the way of minimization elements which do not represent the ideal case participate. Here

$$I(\xi_i > 0) = \begin{cases} 1, & \xi_i > 0, \\ 0, & \xi_i \leq 0, \end{cases}$$

when conditions  $u_i(w^t x_i + w_0) \geq 1 - \xi_i$  and  $\xi_i \geq 0$  are fulfilled. In the formula (8.5) the constant  $\beta$  is the weight considering the bandwidth. If  $\beta$  is not enough, then we allow to arrange relatively many elements in the imperfect position, that is, in the dividing strip. If  $\beta$  is big, then we demand existence of small quantity of elements in the imperfect position, that is, in the dividing strip.

Unfortunately in (8.5), the problem of minimization is rather difficult, because of discontinuity  $I(\xi_i)$ . Instead we will consider a value of minimization  $J(w, \xi_1, \dots, \xi_n) = 1/2 (\|w\|^2 + \beta \sum_{i=1}^n \xi_i)$  with restrictions for all  $i$  in the following form

$$\begin{cases} u_i(w^t x_i + w_0) \geq 1 - \xi_i \\ \xi_i \geq 0 \end{cases}$$

Using Kuhn-Tucker's theorem, we receive

$$L(\alpha) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j u_i u_j x_i^T x_j \rightarrow \max \quad (8.6)$$

provided that  $0 \leq \alpha_i \leq \beta, \forall i$  and  $\sum_{i=1}^n \alpha_i u_i = 0$ .

From the ratio (8.4) we can find  $w = \sum_{i=1}^n \alpha_i u_i x_i$ . The value  $w_0$  can be found, considering that for all  $i$   $0 \leq \alpha_i \leq \beta$  and  $\alpha_i [u_i (w^T x_i + w_0) - 1] = 0$ .

The other idea of the SVM method (in the case when a linear division of classes is impossible), is transition to the space of higher dimension in which such division is possible (see more in Bordes et al. 2005, Burges 1998, Christiani, Shawe-Taylor 2000, Hamel 2000, Rossi, Villa 2006).

For solving the nonlinear classification problem by the linear qualifier it is necessary:

- to design data  $x$  in space of higher dimension by means of transformation  $\phi(x)$ ,
- to find a symmetric discriminant function for data  $\phi(x)$ .

The received nonlinear discriminant function can be written down in the following form

$$g(x) = w^T \phi(x) + w_0.$$

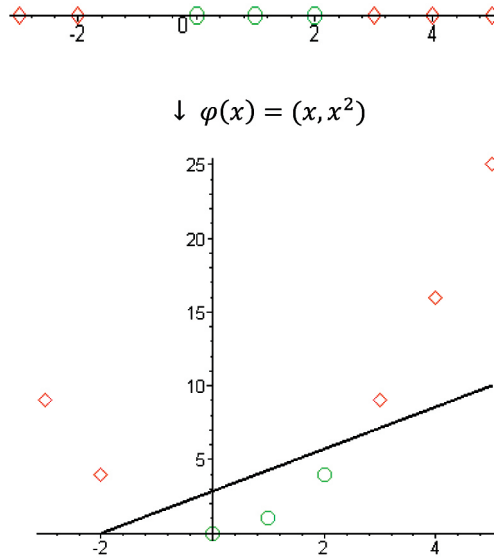
The received in section 8.2 symmetric discriminant function for two-dimensional data  $X$  can take a form

$$g \begin{pmatrix} x^{(1)} \\ x^{(2)} \end{pmatrix} = [w_1 \quad w_1] \begin{bmatrix} x^{(1)} \\ x^{(2)} \end{bmatrix} + w_0.$$

The one-dimensional discriminant function for nonlinear separable data using the function  $\phi(x) = (x, x^2)$  is written as follows

$$g(x) = w_1 x + w_2 x^2 + w_0.$$

The example is shown in Figure 8.5. For transferring data to the space of higher dimension we used so-called kernel functions.



**Fig. 8.5.** An example of linear division of sets upon transition to space of higher dimension

Let's go back to the written above (eq. (8.6)) extremum problem of the method of support vectors given in the following form

$$L(\alpha) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j u_i u_j x_i^T x_j \rightarrow \max.$$

Let's notice that the optimization depends on the formula  $x_i^T x_j$ . If we transfer  $x_i$  to the space of higher dimension using the display function  $\varphi(x)$ , then it is necessary to calculate the similar formula in the space of higher dimension  $\varphi(x_i)^T \varphi(x_j)$ .

The idea of the method requires finding a kernel function  $K(x_i, x_j) = \varphi(x_i)^T \varphi(x_j)$  and to maximizing the following objective function

$$L(\alpha) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j u_i u_j K(x_i, x_j) \rightarrow \max.$$

Let's review the new example and take the kernel function in the form  $K(x, y) = (x^T y)^2$ . It is simple to find out the display  $\phi(x)$  corresponding to the kernel function.

$$\begin{aligned} K(x, y) &= (x^T y)^2 = \left( \begin{bmatrix} x^{(1)} & x^{(2)} \end{bmatrix} \begin{bmatrix} y^{(1)} \\ y^{(2)} \end{bmatrix} \right)^2 = \left( x^{(1)}y^{(1)} + x^{(2)}y^{(2)} \right)^2 = \\ &= \left( x^{(1)}y^{(1)} \right)^2 + 2\left( x^{(1)}y^{(1)} \right)\left( x^{(2)}y^{(2)} \right) + \left( x^{(2)}y^{(2)} \right)^2 = \\ &= \left[ \left( x^{(1)} \right)^2, \sqrt{2}x^{(1)}x^{(2)}, \left( x^{(2)} \right)^2 \right] \left[ \left( y^{(1)} \right)^2, \sqrt{2}y^{(1)}y^{(2)}, \left( y^{(2)} \right)^2 \right]^T. \end{aligned}$$

Thus, the display function can be written in the following form

$$\phi(x) = \left[ \left( x^{(1)} \right)^2, \sqrt{2}x^{(1)}x^{(2)}, \left( x^{(2)} \right)^2 \right].$$

It is important to notice that the choice of kernel function is rather difficult.

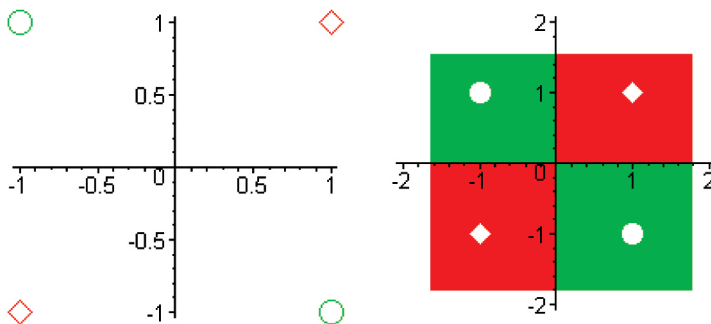
## 8.4. Example

Let's review the example (see more in Joachims 2009):

Class [1]:  $x_1 = [1, -1]$ ,  $x_2 = [-1, 1]$ ,

Class [2]:  $x_3 = [1, 1]$ ,  $x_4 = [-1, -1]$ .

It is illustrated in Figure 8.6.



**Fig. 8.6.** An example of linearly inseparable sets

For creation of nonlinear discriminant function we use kernel function in the following form

$$K(x_i, x_j) = (x_i^T x_j + 1)^2.$$

The display function  $\phi$  corresponding to the kernel function can be written as follows

$$\phi(x) = \left[ 1, \sqrt{2}x^{(1)}, \sqrt{2}x^{(2)}, \sqrt{2}x^{(1)}x^{(2)}, (x^{(1)})^2, (x^{(2)})^2 \right].$$

Further it is necessary to maximize the objective function

$$L(\alpha) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j u_i u_j (x_i^T x_j + 1)^2 \rightarrow \max$$

with restrictions

$$\alpha_i \geq 0, \alpha_1 + \alpha_2 - \alpha_3 - \alpha_4 = 0.$$

Let's rewrite the task in the following form

$$L(\alpha) = \sum_{i=1}^4 \alpha_i - \frac{1}{4} \alpha^T H \alpha,$$

where  $\alpha = [\alpha_1 \quad \alpha_2 \quad \alpha_3 \quad \alpha_4]^T$  and  $H = \begin{pmatrix} 9 & 1 & -1 & -1 \\ 1 & 9 & -1 & -1 \\ -1 & -1 & 9 & 1 \\ -1 & -1 & 1 & 9 \end{pmatrix}$ .

For finding the maximum, we can calculate the partial derivatives with regards to unknown parameters  $\alpha_i$  and equate these derivatives to zero. Then, we will find values of unknown on which the maximum of the objective function is reached.

$$\frac{d}{d\alpha} L(\alpha) = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} - \begin{pmatrix} 9 & 1 & -1 & -1 \\ 1 & 9 & -1 & -1 \\ -1 & -1 & 9 & 1 \\ -1 & -1 & 1 & 9 \end{pmatrix} \alpha = 0.$$

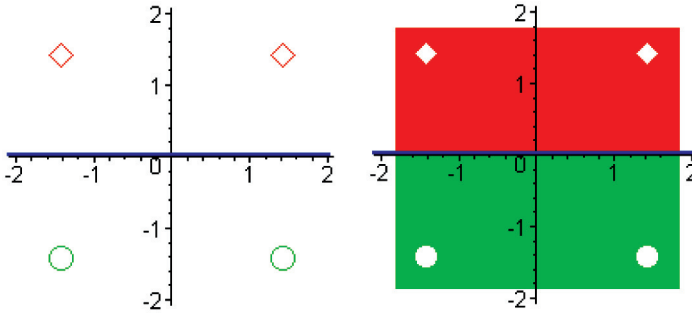
Solving the system equations, we receive  $\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4 = 1/2$ , and

$$w = \sum_{i=1}^4 \alpha_i u_i \varphi(x_i) = \frac{1}{4} (\varphi(x_1) + \varphi(x_2) - \varphi(x_3) - \varphi(x_4)) = [0 \quad 0 \quad 0 \quad -\sqrt{2} \quad 0 \quad 0]$$

and, at last, the nonlinear discriminant function can take the following form

$$g(x) = w\varphi(x) = \sum_{i=1}^6 w_i \varphi_i(x) = -\sqrt{2} (\sqrt{2} x^{(1)} x^{(2)}) = -2x^{(1)} x^{(2)}.$$

The result is shown in Figure 8.7.



**Fig. 8.7.** An example of linearly inseparable sets after using the kernel function

In conclusion we give a few examples of the most widespread kernel functions used for division of classes:

- The polynomial homogeneous kernel  $K(x_i, x_j) = (x_i^T x_j)^d$ .
- The polynomial heterogeneous kernel  $K(x_i, x_j) = (x_i^T x_j + 1)^d$ .
- The radial basis function (RBF kernel)  $K(x_i, x_j) = \exp\left(-\|x_i - x_j\|^2 / 2\sigma^2\right)$ .
- The sigmoid kernel  $K(x_i, x_j) = \tanh\left(\left(x_i^T x_j + 1\right)\right)$ .

## 9. Visualization of multidimensional data

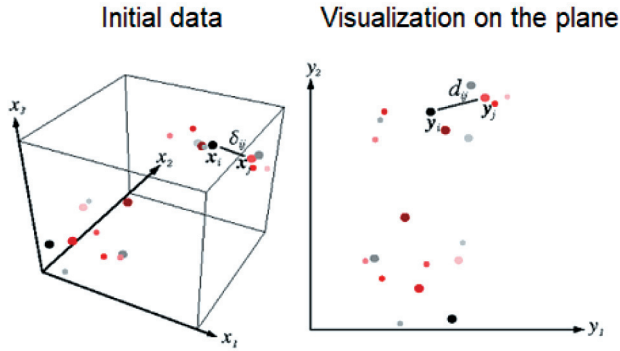
A person analyzing two-dimensional, or, as a last resort, three-dimensional data, when working with larger dimensions data may encounter some problems. These days, the majority of information which demands the analysis, has a significantly big dimension. So, what to do in such cases? The tools supporting the researchers analyzing the multidimensional data sets, are data visualization methods, that enable a consolidation and a reduction of the data dimension which in turns allows to represent them in two or three-dimensional spaces (see, for example, Mills et al. 2013). This paragraph is devoted to some of such methods.

### 9.1. Multidimensional scaling technic

It often happens that the information is presented in the form of a square matrix that contains distances between the examined objects. At the intersection of  $i$ -th row and  $j$ -th column in such a table there is a value specifying difference (or similarity) of the two objects  $i, j$ . This form of presenting information is characteristic for different researches especially when the similarity or distinction, in some systems of objects or concepts, is examined.

Thus, initially no coordinate in multidimensional space is combined with the object and the presentation of such data in the form of some geometrical interpretation is rather difficult. The problem of *multidimensional scaling* (see, for example, Kruskal, Wish 1978, Gorban et al. 2008) is to design distribution of data in space so that distances between individual objects correspond to the values from the distance matrix. The coordinate axes arising in this way can be interpreted as some factors which value defining objects distinction among themselves, like shown in the section 2 the principal components analysis (PCA). The main goal is to find the map that estimates the original distances and tells where the points are located. In that case, we obtain coordinates pairs corresponding to individual objects and the possibility their visualization in two or three-dimensional space (see Fig. 9.1).





**Fig. 9.1.** An Illustration of multidimensional scaling

In the linear method of multidimensional scaling the method of matching points is the main component, but it is not applied to the initial matrix of distances. The distance matrix must be *double centered matrix* so the averages in the rows and in the columns of the matrix are zero. Twice centered matrix is unambiguously calculated based on the initial similarity matrix (there are more options presented in Olshen, Rajaratnam 2010). After that, there is an opportunity to define a dimension of the space providing *exact* reproduction of the distance matrix or to define an effective dimension of the designed space of points which will provide the estimated representation of the distance matrix of with the given accuracy.

In nonlinear methods the space dimension is set initially, and, by means of gradient methods the functionality of quality describing the distortion measure of the distance matrix is optimized.

Going forward, we can formalize the problem of definition. So, let the points  $x_1, x_2, \dots, x_n$  of the  $k$ -dimensional space be given. Let us denote by  $\delta_{i,j}$  the distance between two points  $x_i$  and  $x_j$ . Let's find points  $y_1, y_2, \dots, y_n$  in the space of smaller dimension (2 or 3) so that the distance between them can be equal  $d_{i,j}$  and are put according to  $\delta_{i,j}$ . It would be desirable to obtain direct compliance  $d_{i,j} = \delta_{i,j}$ , but in the case of transition to the space of smaller dimension it is hardly possible to receive such equality. But still this problem needs to be solved, and, one of possible methods of finding the dependence between  $d_{i,j}$  and  $\delta_{i,j}$  is to minimize the selected objective function. In such a task it is possible to use minimization of the mean square mistake as follows (see more in Gorban et al. 2008, 2016)

$$J(y) = \frac{\sum_{i < j} (d_{i,j} - \delta_{i,j})^2}{\sum_{i < j} \delta_{i,j}^2}.$$

The minimization of the objective function is a difficult task. The gradient descent method, as it is shown below, can be used to solve it.

$$\nabla(\delta) = \frac{2}{\sum_{i < j} \delta_{i,j}^2} \sum_{j \neq k}^k (d_{k,j} - \delta_{k,j}) \frac{(y_k - y_j)}{d \sum d_{k,j}}$$

Coordinates of  $y_1, y_2, \dots, y_n$  points are selected to receive the greatest change.

It is worth noticing that the multivariate data distribution displayed on the plane does not resolve all issues of input data visualization. The idea to display, on the two-dimensional card, not only points of data, but also the various information accompanying input data, for example, the location of points in initial space, density of different subsets, other information about subsets with continuously distributed in the initial space is absolutely natural. Everything results in the idea of more effective use of all primary information for displaying both quantitative, and qualitative information.

At last, on the deployment of data to the two-dimensional plane, it would be desirable to have the opportunity to arrange, on the two-dimensional plane the data which did not participate in display setup. It would allow, on the one hand, to use the received picture for creation of different expert systems and solving problems of pattern recognition, with the other – to use it for recovery of data with gaps. As a result we come up with the idea of using an auxiliary object called *a map* for visualization of data and extraction of information. This object stands for a limited two-dimensional nonlinear representation enclosed in multidimensional space of data so that it can serve as a data model that has the form and the arrangement of such diversity that it has to reflect the main features of distribution of the data points set.

A simple example of data mapping is the plane of the first two main component. As it has been already noted above, among all two-dimensional planes enclosed in a space, the plane of the first two main components serves as an optimum screen on which it is possible to display the main proper patterns for data. As another, a simpler (but not an optimum) map it is possible to use any coordinate plane of any two chosen coordinates.

## 9.2. Kohonen self-organizing maps (SOM)

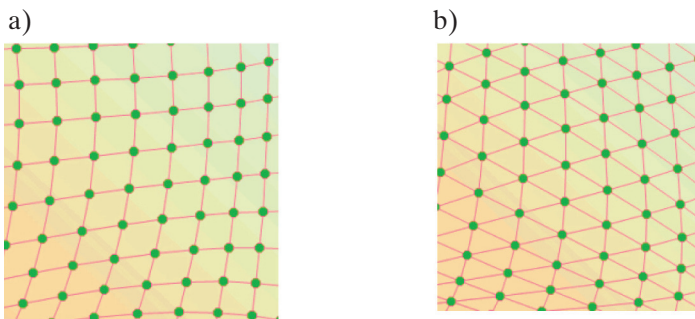
Our goal, as mentioned above, is to project available data onto a lower dimension space, maintaining the distance between the available points (objects).

Kohonen maps (see more in Kohonen 1990, 2001, 2013, Kohonen, Somervuo 1998, Pastunkov, Prokofiev 2016, Penn 2005, *Scholarpedia Kohonen network...*, *Self-organizing map...*) make cartography from the multidimensional entry into the grid of one- or

two-dimensional nodes with use of ideology of neural networks. Important feature of this cartography is preservation of topology between input data and neurons into which they are projected. Kohonen's self-organizing maps represent a neural network based on a greedy algorithm which requires no training. It is important to notice that the idea of SOM is based on the brain functioning. Activation of the neuron by perception of information leads to excitation of sites in other parts of the brain.

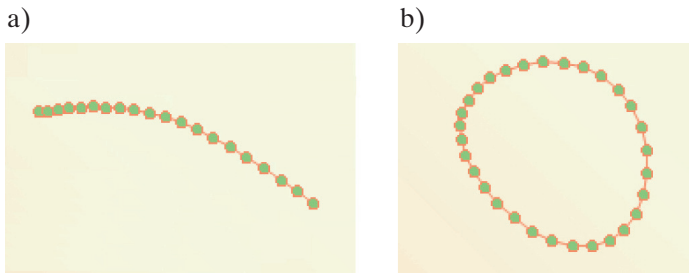
Let's pass to the detailed statement of ideology of SOM. The neural network architecture offered by Teuvo Kohonen (Kohonen 1982) for the automatic clustering (classification without teacher), is based on ingenious use of information about the relative position of the neurons forming the grid. The signal reaches such a neural networks simultaneously on all neurons. The weight of the respective synapses is interpreted as coordinates of incoming nodes, and, the output signal is created according to the "the winner takes away everything" principle, i.e. the non zero output signal belongs to a neuron, that is the next (in sense of scales of synapses) relative to the object given at the input. In the course of training the synapses weights are adjusted so that nodes of the grid are located in the places of local condensations of data. In such a way it can describe a cluster structure of data. On the other hand, communications between neurons correspond to the neighborhood relations between the corresponding clusters in the space of data (more details are in Kohonen 1982, 2001, Manukyan et al. 2011).

Initially SOM represents the grid of the nodes connected among themselves by communications. Kohonen considered two options of nodes connection – with the rectangular and hexagonal grid (see Fig. 9.2). In the rectangular grid each node is connected with four next nodes, and in hexagonal – with six next nodes.



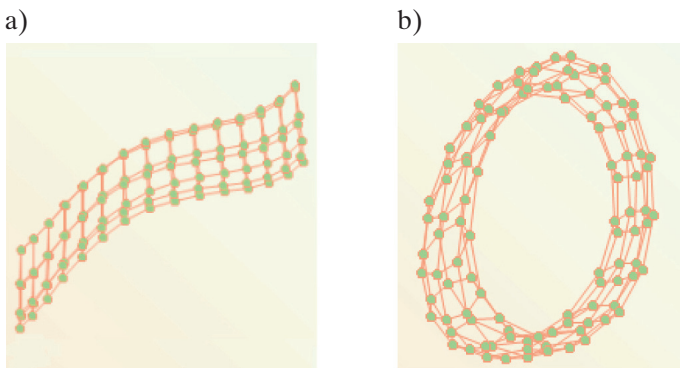
**Fig. 9.2.** Examples of use of two-dimensional grids of neurons in the Kohonen SOM:  
a) neurons in the rectangular grid – each neuron has 4 closest neighbors;  
b) neurons in the hexagonal grid – each neuron has 6 closest neighbors

It is important to notice that among the most popular topologies of Kohonen's neural networks there can be mentioned two cases shown in Figure 9.3.



**Fig. 9.3.** One-dimensional Kohonen's networks:  
a) linear layer; b) circular layer

Next two cases shown in Figure 9.4 (see more in Kohonen 2001, 2013).



**Fig. 9.4.** Two-dimensional Kohonen's networks:  
a) cylindrical layer; b) toroidal layer

### 9.2.1. Initialization of the map of Kohonen

SOM is the cornerstone referring to the coherent structure of neurons competing among themselves about the influence of the signal. The weight  $w_{ij}$  is delivered to each neuron in compliance with the number of a scale which is equal to the dimension of the input signal.

There are several ways of preliminary initialization of the map. It assigns values to all scale vectors referring to neurons with the weight  $w_{ij}$  as follows:

1. selection of all coordinates as random numbers;
2. assignment a random sample from input data to the initial vector;
3. selection of vectors from the set of main components (PCA) of input data.

It is necessary to notice that after initialization each neuron does not move on the map, i.e. the vector of  $p$  does not move during the entire training.

### 9.2.2. The training algorithm

The algorithm of training is made by iterations.

Let denote by  $t$  an index of an iteration step. Let's assume that initialization has a number of iteration 0. Further, the following operations are carried out as follows: We choose the accidental vector  $x(t)$  from the set of input values.

We find a distance to all scale vectors of all neurons on the map. For this operation some measure can be chosen. Not to mention that the mean square deviation is the most often used. We look for a neuron which is the closest to the input value  $x(t)$

$$d(x(t), w(t)_{\text{vM}}) < d(x(t), w_{ij}(t)),$$

where  $w(t)_{\text{vM}}$  is the weight of the winner neuron (BMU(best matching unit), Winner)  $w_{\text{vM}}(t)$ ,  $w_{ij}(t)$  – the vector of weight,  $d(x(t), w_{ij}(t))$  – the certain measure of the distance, for instance mean square deviation (MSD).

It is important to notice that if several neurons meet the above-stated condition, then the winner neuron is chosen in a random way (see Fig. 9.5). After that the nodes begin to move in the space. However, the node does not move alone, but takes with itself a certain quantity of the next nodes from some vicinity on the map. Let's explain it: if the neighborhood radius is equal 1, then together with the node four of his neighbors are moved on the map, in the case of rectangular grid. For hexagonal grid 6 neighbors are transferred with the selected node.

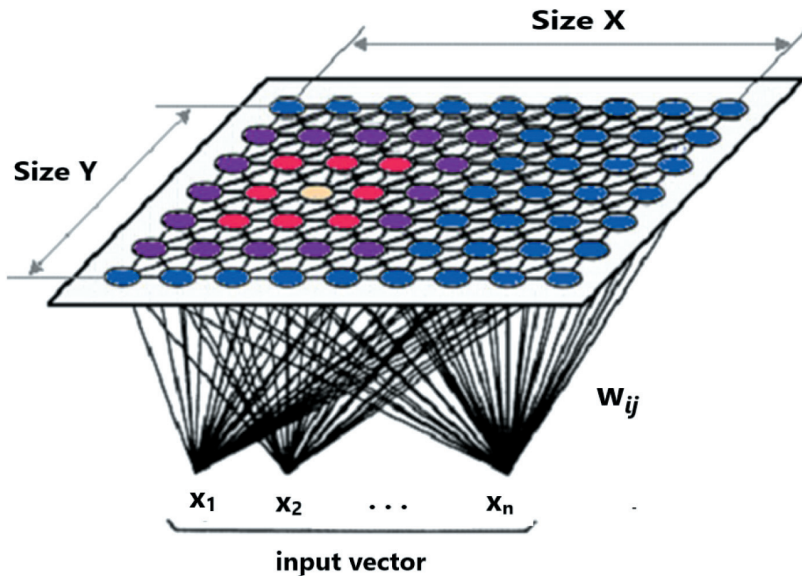


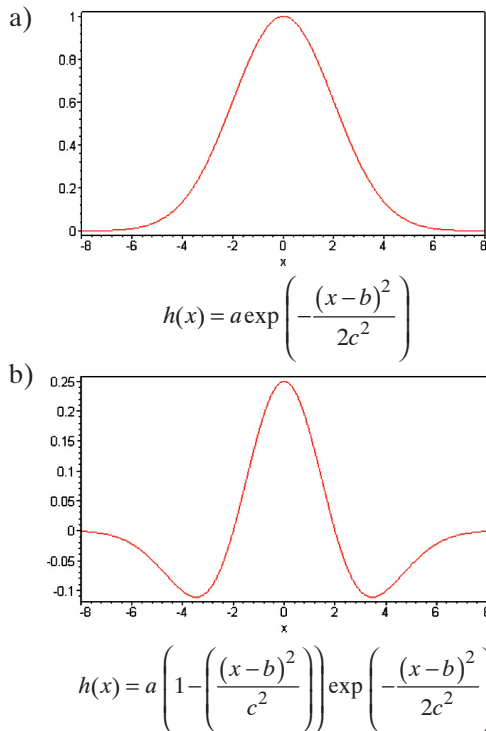
Fig. 9.5. Kohonen's self-organizing map (SOM)

In the training process of the map two phases are distinguished – the organizing phase (*ordering*) and the convergence phase (fine-tuning). At the first phase neighborhoods of great values are chosen, and, the movement of nodes has collective character. As a result the map “organizes” itself and roughly reflects the structure of data. At the phase of fine tuning the radius of the vicinity is equal 1–2 and individual weights of nodes are already adjusted.

It is necessary to mention one more property of the movement nature. Usually it is adjusted in such a way that among all moving nodes the strongest one is center dislocated – the next to the point of the data. The others are tested in a way that the smaller shifts are further from the central node.

Character of the movement is built up by the so-called fading functions of the neighborhood (neighborhood function). If all neighbors from the environment have equal shifts, then such a function of the neighborhood is called bubble and for it the bigger number of movements and less smooth grid is characteristic in the training.

Let’s consider in more detail a definition of the measure of neurons neighborhood and the change of neurons weight on the map. We choose the neighborhood measure as the  $h(t)$  function. Usually, as the  $h(t)$  function the Gaussian function is used (see Fig. 9.6).



**Fig. 9.6.** Examples of the neighborhood functions: a) Gauss’s function; b) Mexican hat (rated second differential coefficient of Gauss’s function)

Let  $h(t)$  be Gauss's function, then "the neighborhood measure" between neurons  $M_{i,j}$  and  $M_{v,\mu}$  will take the following form

$$h_{i,j}^{v,\mu}(t) = \alpha(t) \exp \left( - \frac{\|r_{i,j} - r_{v,\mu}\|^2}{2\delta^2(t)} \right),$$

where:

$0 < \alpha(t) < 1$  – the training factor monotonously decreasing with each subsequent iteration (that is defining vector values referring to the winner neuron weight and its neighbors; the larger step, the less clarification is;  $t$  is an index of iteration step),

$r_{i,j}, r_{v,\mu}$  – coordinates of the  $M_{i,j}(t)$  and  $M_{v,\mu}(t)$  nodes on the map,

$\delta(t)$  – monotonously decreasing factor regulating the number of neighbors depending on an iteration step.

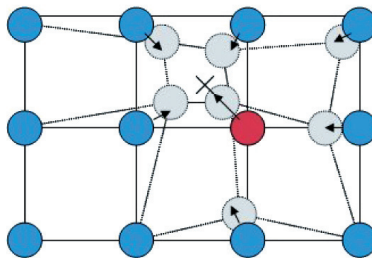
Parameters  $\alpha$ ,  $\delta$ , and their decrease nature, are defined by the expert way. The easier way to determine the neighborhood function is set as follows:  $h_{i,j}^{v,\mu}(t) = \alpha(t)$  if  $M_{i,j}(t)$  is in the vicinity of  $M_{v,\mu}(t)$  with in advance set radius, and 0 – otherwise.

The  $h(t)$  function is equal  $\alpha(t)$  for the winner neuron and decreases with removal from it.

The following phase is a calculation of the map error. The weights vectors are changed according to the formula

$$m_{i,j}(t) = m_{i,j}(t-1) + h_{i,j}^{v,\mu}(t)(x(t) - m_{i,j}(t-1)).$$

Thus, all nodes which are neighbors of the winner neuron, approach the considered winner as shown in Figure 9.7.



**Fig. 9.7.** Scheme of the movement of neurons

At last, the choice of the condition for the stop of map formation process is taken under consideration.

For determining the criterion for the stop of the map error, for example, in most cases we use the arithmetic average distance between observations and vectors of weight corresponding to the winner neuron is in most cases used. It can be formulated as follows

$$\frac{1}{N} \sum_{i,j} \|x_{i,j} - w_{i,j}\|,$$

where  $N$  is a number of input data elements.

The algorithm repeats the certain number of steps. On the first phase the chosen number of steps is about a thousand, on the second – ten thousand (it is clear that the number of steps can strongly change depending on tasks).

Let's note that the offered algorithm does not use any criterion of optimization. However it is clear that the average distance from each point of data to the closest node of the card will decrease, at least. The average square of such distance serves as quality criterion of the constructed map. Usually several tens maps for which receives the best result according to the mentioned criterion, are under construction.

As a result of the algorithm action the two-dimensional grid of the nodes placed in multidimensional space (map) is under construction. To represent the input data collocations different types of illustrations are used. For instance it can be the coloring of the map when color reflects distance between nodes.

In the sections below several examples are given.

## 9.3. Examples

### 9.3.1. Showing similarity of objects

Let the random points of the plane be given as it was shown in Figure 9.8.

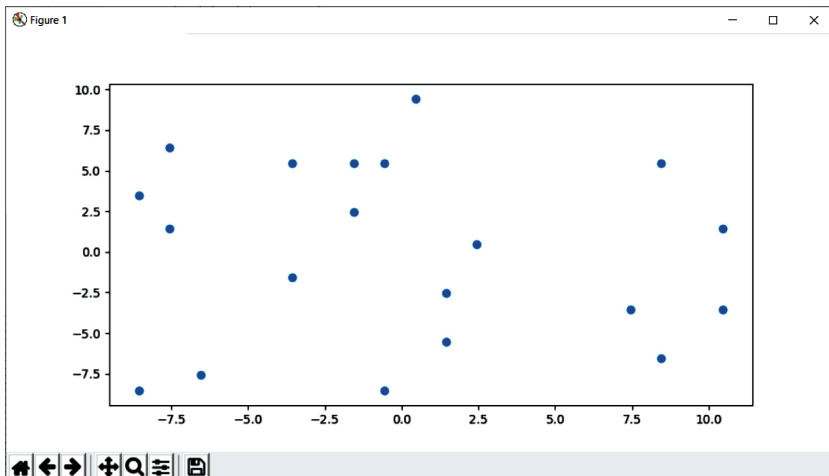


Fig. 9.8. The random generated points



The each point has coordinates which can be used to determine the matrix of distances between points. The obtained matrix is next randomly disturbed. The matrix is further used to establish new points in the multidimension scaling procedure. The received new points are shown in Figure 9.9.

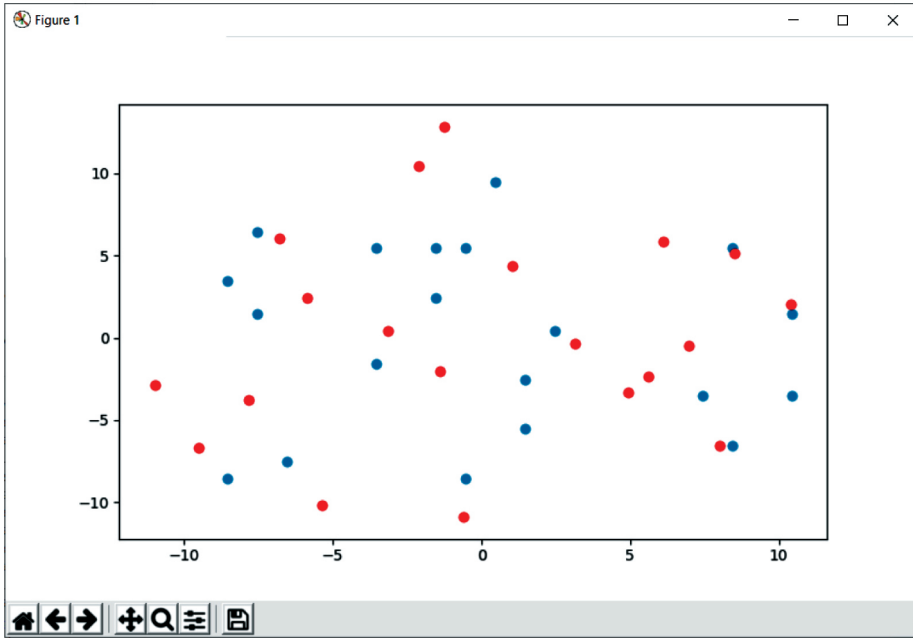


Fig. 9.9. The random generated blue points; red points are obtained from multidimension scaling procedure

### 9.3.2. Showing similarity of European countries

On the basis of Eurostat data, indicators showing the level of social life in European countries were selected. The following five indicators were taken into account: employment rate from 2016 and 2017, gross domestic expenditure on R&B (on the year 2016 and 2017), poverty and exclusion risk indicator (on the year 2016 and 2017), number of people living in households with very low work intensity of those years and expenditure on pensions (on 2016 year). In total, the data counted seven variables from 34 countries. In Figure 9.10, the similarity structure was shown depending on the choice of the distance measure between selected countries. The data are transform from seven-dimensional space to two-dimensional space.

It is chosen euclidean, cosine, canberra and mahalanobis distances.

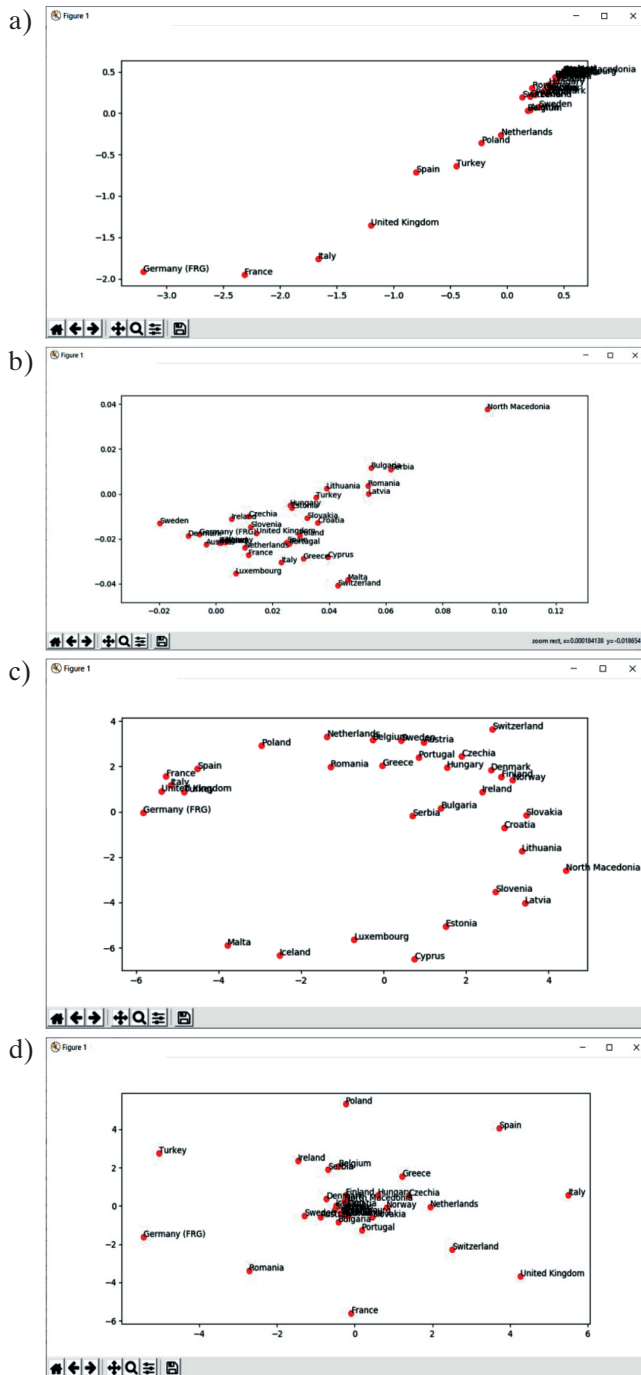


Fig. 9.10. The similarity of European countries: a) euclidean distance; b) cosine distance; c) canberra distance; d) mahalanobis distance

### 9.3.3. The world map of poverty

On the website (<http://www.cis.hut.fi/nnrc/worldmap.html>) of Helsinki University of Technology Laboratory of Computer and Information Science the example of SOM use for creation of the certain composite economic indicator which is used for coloring the normal map is given. As a result, the countries with the similar economic situation are represented on the map by colors with similar shades.

For creating of the map 39 features (indicators) describing different factors of life quality were considered. Among them it can be listed for example, the health system level, educational services, quality of the power supply and others. Countries for which the available values of indicators are similar create clusters that were automatically coded on the map in corresponding colors. As a result of this process, each country of the same color description characterizes the same poverty level – from yellow shades for the safe countries to violet and blue shades for poverty countries as it is shown in the Figure 9.11.

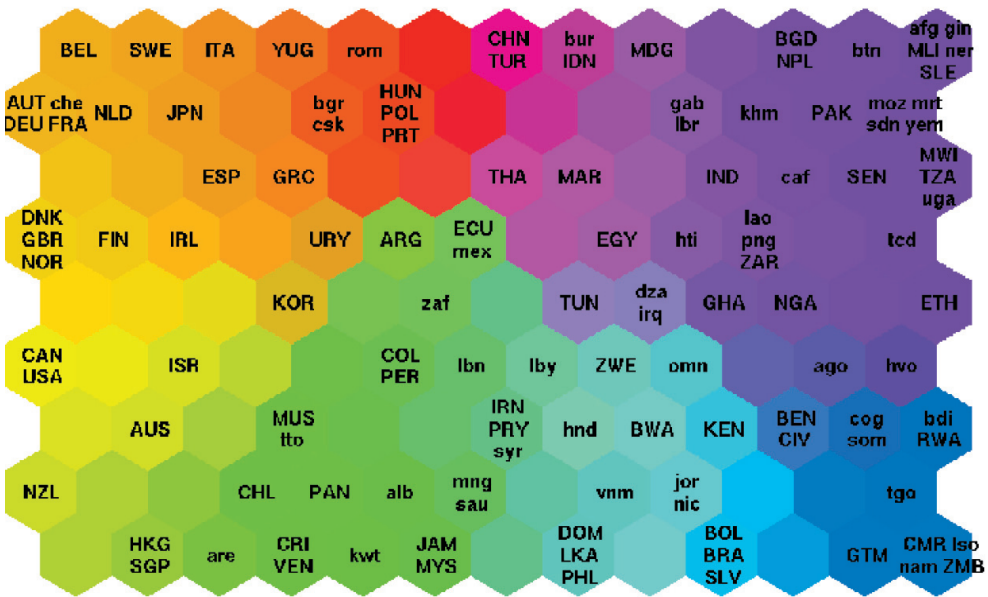
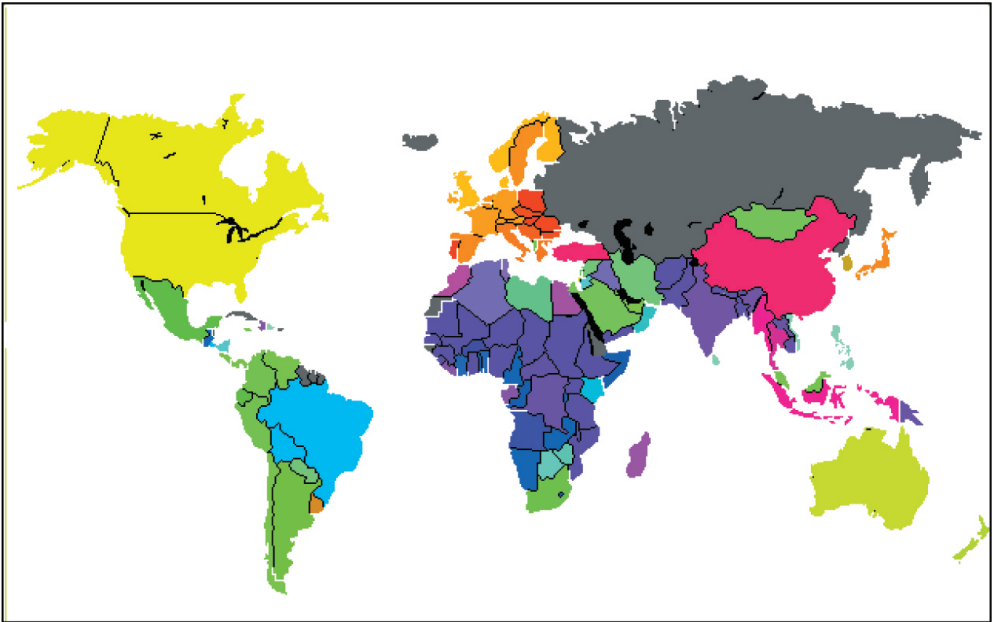


Fig. 9.11. Distribution of colors on the map of the world poverty

In Figure 9.12 each country is painted in color according to the poverty level. The countries for which (for the period of the research) the necessary statistics was either incomplete, or unavailable are painted in gray color.



**Fig. 9.12.** The map of the world poverty

### 9.3.4. The traveling salesman problem

Let's review one more example Kohonen networks use to solve the traveling salesman problem.

The purpose of the traveling salesman problem consists in finding the shortest route that the salesman moves starting from the one city, passing through all cities on the map and returning to the initial city. Formally, we have to find the shortest Hamilton cycle of the nondirectional weighed graph containing in quality of the node – the city, and the distance between two cities is the weight of the edge connecting the corresponding nodes. Here we will try to find the approximate solution of this task with use of self-organizing maps of Kohonen. Input data consist of coordinates (X and Y) all cities on the sales map. The topology of Kohonen's network is selected as the circular layer.

The following three figures (see Fig. 9.13–9.15) show the application of SOM to the solution of the traveling salesman problem. The initial sales maps is given in the first figure (Fig. 9.13). On the last figure (Fig. 9.15) the received route of the salesman is illustrated. The received way is the shortest among all possible routes connecting all cities on the sales map. Intermediate results are shown in Figure 9.14.

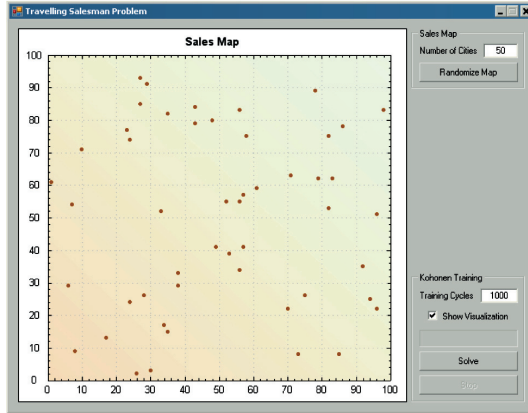


Fig. 9.13. The sales map

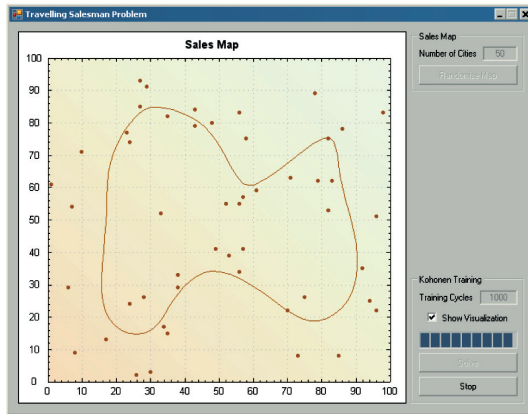


Fig. 9.14. The route of the movement after 100 iterations

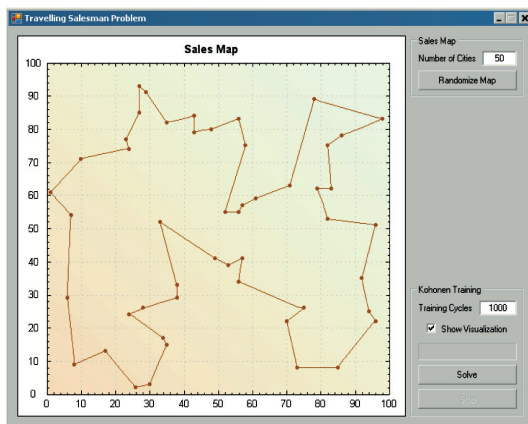
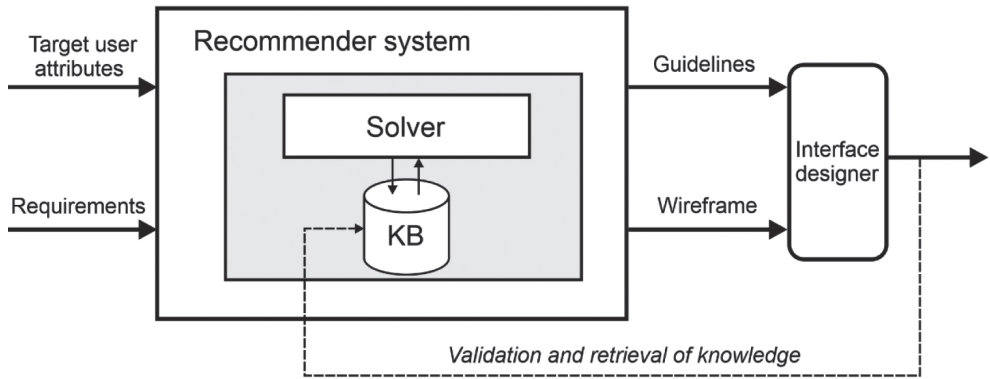


Fig. 9.15. Solution of the traveling salesman problem

## 10. Recommender systems

The purpose of the recommending system (see, for example, Basu et al. 1998, Ricci et al. 2011, Resnik, Varian 1997, Schafer et al. 1999) is to create an important recommendation for users regarding a set of items or products (we will use further the term subject), which may be of interest them. It can be for instance an offer referring to books, goods or movies. Developing such recommendations depends on the area and specific characteristics of the data. For example, the Netflix website provides an opportunity of rating movies on the scale from 1 to 5 (very bad, bad, average, good, very good). Such records for data source allow to implement interaction between users and elements of the online request. Besides, using such attributes as demographic data and the product description, the system can have access to the specific user and point in a specific profile. The recommending systems differ in methods of the analysis of the data sources for a communication research between users and elements which can be used for identifying well correlated couples the client – the item. Such analysis system as collaborative filtering (see in Brusilovsky et al. 2007, Cohen, Fan 2000, *Collaborative filtering...*, Hill et al. 1995) and content-filtering (see for instance Kuppussamy, Aghila 2011, Marlin 2004, Sawar et al. 2001) on the basis referring to goods or services and first and foremost to their attributes are a cornerstone of recommending systems. Besides, there are hybrid methods (see for instance Brusilovsky et al. 2007, Chakrabarti 2004, Marlin 2004, Ricci et al. 2011) which represents an attempt of consolidation both these designs. The general structure of the recommending system can be shown a follows (see Fig. 10.1).

Obtaining recommendations from reliable sources is one of the most important components of natural decision making process. The growing consumption supported by the development of Worldwide Network leads to the fact that more buyers are offered a wider range of services while sellers face the problem of personalized advertising campaigns. Recommender systems are evolving to meet the needs of buyers and sellers using the automation of recommendations based on data analysis.



**Fig. 10.1.** The general structure of the recommending system

The term “collaborative (joint) filtering” (Chakrabarti 2004) was entered in the context of the first commercial recommending system which was developed to address the recommendation of news to a certain circle of users. The motivation assumed that the user would not receive uninteresting documents. Joint filtering analyzes data on the use of goods or services items by different users to identify a well-chosen pair: user – object. According to the filtering methodology that leads to information sources, search recommendations are not “joint” in the sense that offers presented by users clearly do not use the information of all users, and focus on attributes of items (goods or services).

The first methods for the recommendation system were based on simple statistics of correlation and forecast modeling.

Further research was caused by general availability of data on the Internet, and interest in the growing popularity of electronic commerce. The surge in interest in this problem was caused by the Netflix, a company specializing in video and DVD streaming, has raised interest in this problem providing free access to a large data set containing 100 million ratings, about half a million users, and several thousand movies. The company announced an open tender for the best algorithm of collaborative filtering (see Fig. 10.2).

Now, the recommendation systems attract the attention of active research from various fields in which sections of mathematical statistics, machine training, data mining and so on are crossed. Appendices of methods of the recommendation systems are performed in different areas, beginning from the web pages recommending music, books, movies and other consumer goods ending with the interpretation of political elections.

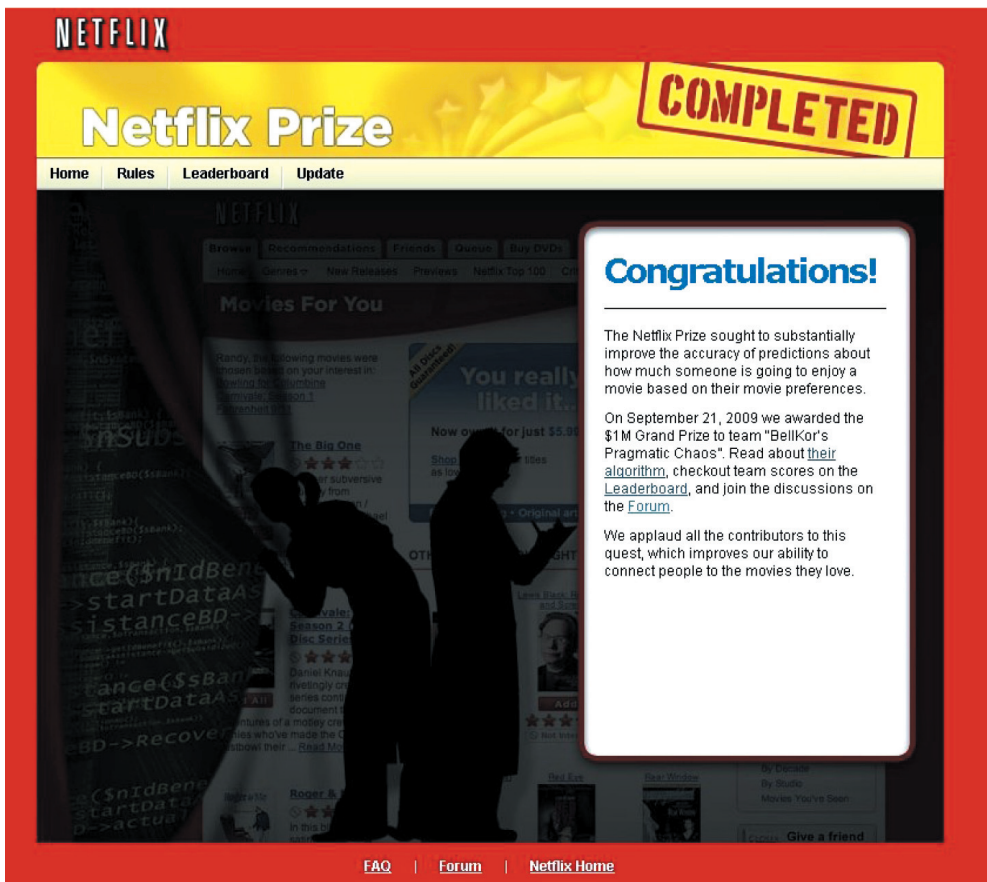


Fig. 10.2. Netflix message on the end of a tender

## 10.1. General structure of recommendation system

The general structure of the recommending system is provided in Figure 10.1. The known preferences of the user are presented in the matrix form composed of  $N$  users and  $M$  items (products, elements, services – see Fig. 10.3) where each cell shows the rating of the item  $i$  connected with the user  $u$ .

The matrix of ratings is very rarefied as most of users fills only several points. Feature is that the sparseness of a matrix is understood as not traditionally mathematical definition (a large number of a matrix coefficients is equal to zero), and an absence of any information in most of cells. A problem of recommendatory system is to predict user's rating and fill the blank cells.



		<i>Items</i>					
		1	2	...	<i>i</i>	...	<i>m</i>
<i>Users</i>	1	5	3		1	2	
	2		2				4
	:			5			
	<i>u</i>	3	4		2	1	
	:		5			4	
	<i>n</i>			3	2		
		<hr/>					
	<i>a</i>	3	5		?	1	

**Fig. 10.3.** A rating matrix  $r_{ui}$  connecting the user  $u$  with an object  $i$

The set of approaches to the recommending systems can be classified as follows:

- Joint or collaborative filtering (CF – Collaborative Filtering): in the CF systems, it is recommended to assign the user the elements on the basis of the last collective ratings of all users.
- Content-oriented recommendations (CB – Content Based Recommendation): these approaches recommend services similar to maintaining services that were pleasant to the user in the past or that correspond to the pre-defined attributes of user preferences.
- Hybrid approaches: these methods combine the essence of the given above approaches.

### 10.1.1. Collaborative filtering

Joint (collaborating) filtering (CF) system works by implementing the feedback using the similarity in the rating of behavior among several similar users. In other words, CF methods are based on proximity of neighbors and models (see in Poncet et al. 2008, Rocchio 1971, Cohen, Fan 2000, Ricci et al. 2011, Symeonidis et al. 2006, Tian, Kwok-Wai Cheung 2003).

Let's consider simple model of the recommendation system. The proximity of neighbors is a basis of these methods. The members of the user's group are extracted based on their similarity to the active user, and, the weighed combination of their ratings is used for obtaining the user's predictions.

Such algorithms are based on the following steps:

1. Arrange the set of weights for all users with similar preferences to the active user.
2. Select  $k$  users which have the greatest similar preferences to the active user – the nearest neighbors.
3. Calculate the forecast from the weighed combination of the chosen neighbors.

In the first step, the weight of  $w_{a,u}$  is a similarity measure between the user  $u$  and the active user  $a$ . The most often used measure of similarity is the Pearson correlation coefficient between the ratings of two users

$$w_{a,u} = \frac{\sum_{i \in I} (r_{a,i} - \bar{r}_a)(r_{u,i} - \bar{r}_u)}{\sqrt{\sum_{i \in I} (r_{a,i} - \bar{r}_a)^2} \sqrt{\sum_{i \in I} (r_{u,i} - \bar{r}_u)^2}},$$

where  $I$  is a set of elements evaluated by users,  $r_{u,i}$  the rating of the service  $i$  by the user  $u$  and  $\bar{r}_u$  is the average assessment of the user  $u$  rating.

The second step, was discussed in the section 5.4 where clustering methods were considered.

The third step, called forecast, as a rule, the predict values  $p_{a,i}$  are calculated as the weighted average deviations from neighbors as follows

$$p_{a,i} = \bar{r}_a + \frac{\sum_{u \in K} (r_{u,i} - \bar{r}_u) \times w_{a,u}}{\sum_{u \in K} w_{a,u}},$$

where  $\bar{r}_a$  is the average rating for active user  $a$ ,  $r_{u,i}$  is the rating of service  $i$  by the user  $u$ ,  $w_{a,u}$  – similarity between users  $a$  and  $u$ , and  $K$  – neighbors or the set of users, the closest to the active user  $a$ .

Besides, if we consider the ratings of users as vectors, then it is possible to estimate their similarity based on a cosine of the angle between them

$$w_{a,u} = \cos(\vec{r}_a, \vec{r}_u) = \frac{\vec{r}_a \cdot \vec{r}_u}{\|\vec{r}_a\|_2 \|\vec{r}_u\|_2} = \frac{\sum_{i=1}^m r_{a,i} r_{u,i}}{\sqrt{\sum_{i=1}^m r_{a,i}^2} \sqrt{\sum_{i=1}^m r_{u,i}^2}}.$$

Let's notice that empirical research shows that Pearson's correlation usually works better. There are also other measures of similarity, including a mean square mistake, entropy and so forth.

**The choice of an item based on collaborative filtering:** With the advent of the Internet, the use of traditional methods for obtaining data applying to millions of users and objects, is not effective due to computational complexity. As an alternative, the scheme of collaborative filtering is dedicated to an element-to-element relations in which compliance is determined not by similar users, but by rating of similar items. In practice, this approach leads to more quickly reacting recommendation system and often leads to improvement of predicting the user behavior.

At such approach, the similarity between couples of items  $i$  and  $j$  is calculated using the Pearson correlation

$$w_{i,j} = \frac{\sum_{u \in U} (r_{u,i} - \bar{r}_i)(r_{u,j} - \bar{r}_j)}{\sqrt{\sum_{u \in U} (r_{u,i} - \bar{r}_i)^2} \sqrt{\sum_{u \in U} (r_{u,j} - \bar{r}_j)^2}},$$

where  $U$  is a great number of all users who estimated both elements  $i$  and  $j$ ,  $r_{u,i}$  is the rating of the element  $i$  by the user  $u$ , and  $\bar{r}_i$  is the average rating of the element  $i$  among users.

Now, the rating of the element  $i$  by the user  $a$  can be predicted by means of the weighed assessment as follows

$$P_{a,i} = \frac{\sum_{j \in K} r_{a,j} w_{i,j}}{\sum_{j \in K} |w_{i,j}|},$$

where the neighborhood of item  $i$  is a set of  $k$  elements that are the closest to  $i$  due to  $a$ .

Let's notice that at measurement of similarity among users, elements which were evaluated by all rating marks (from 1 to 5), are not so useful as less widespread elements. To consider this fact we use a concept of the user return frequency which is calculated as follows  $f_i = \log(n/n_i)$ , where  $n_i$  is the number of users who evaluated an element  $i$  from total number of  $n$  users. To apply the user return frequency while using the similarity based on CF, the original rating of  $i$  is multiplied by  $f_i$ . The basis of this approach is that elements which are good and very good, are estimated more often than others.

In the case of big arrays of users and items, outstanding performance was shown by the latent models and factorization of matrix models. Unlike the methods based on localization of close neighbors who generates recommendations on statistical concepts of similarity among users or among elements, the latent factor models are based on that the similarity between users and items is induced by some hidden data structures of smaller dimension. For example, the rating which the user gives to the movie, most likely, depends on several implicit factors, such as the user taste, the user favorite actors or favorite directors of the movie, etc. Matrix factorization is called the class of successful CF models with the hidden factors where users and items are presented in the form of unknown vector functions (vector columns)  $w_u, h_i \in \mathfrak{R}^k$  in  $k$  hidden dimensions. The result of the product  $w_u^T h_i$  is an approximation of a matrix relating to the known ratings  $r_{u,i}$ . To find the  $w_u, h_i$  the least-squares method is traditionally used as follows

$$J(W, H) = \sum_{(u,i) \in L} (r_{u,i} - w_u^T h_i)^2,$$

where  $W = [w_1 \dots w_n]^T$  is a searched matrix of the size  $n \times k$ ,  $H = [h_1 \dots h_m]$  is a searched matrix of size  $k \times m$ , and the set of couples user – element for which ratings  $r_{u,i}$  are known is  $L$ . In a trivial case when all ratings of couples user – element are known, the objective function takes a form

$$J(W, H) = \|R - WH\|_{\text{fro}}^2$$

where  $R$  designates  $n \times m$  matrix of completely known couples user – item and  $\|\bullet\|_{\text{fro}}$  – Frobenius's norm defined as follows  $\|A\|_{\text{fro}} = \sqrt{\text{tr}(A^T A)}$ . Thus finding the minimum of this task is given by singular value decomposition (Singular Value Decomposition – SVD; see more for instance in Sawar et al. 2000). However, in practice, when the majority of items ratings by the user is unknown, a global optimal solution cannot be received in an explicit form. Therefore optimization of not convex objective function of  $J(W, H)$  is required. In this case the objective function is represented by a special form of the weighed losses, i.e.

$$J(W, H) = \|S \otimes (R - WH)\|_{\text{fro}}^2,$$

where  $\otimes$  designates the element-by-element product, and  $S$  is a binary matrix with the coefficients equal to one for the known couple user – element of  $L$ , and to zero otherwise. The standard decision assumes the use of gradient methods or procedures,

similar to iterative modification of the least-squares method when  $H$  is estimated at the fixed  $W$ , and, on the contrary, when  $W$  is estimated at fixed  $H$  until the criterion of convergence is reached. Similar approach was considered in the section 1.2 devoted to the least-squares method.

After estimating  $W$  and  $H$ , their product  $WH$  provides the buyer with a recovery rating matrix from which recommendations can be evaluated directly.

Different choice of objective function, regularizations and additional restrictions of models evoke a great interest in matrix factorization methods. It can be claimed that for discrete ratings, the square of errors is not the most natural method.

Another class of methods uses matrix factorization with restriction of nonnegativity imposed on  $W$  and  $H$ . The rating of each user behavior can be considered, by means of manifestation different roles, for example, by presence at groups of the users connected by interests or communication that leads to the objective function as follows

$$J(W, H) = \sum_{(i,u) \in L} r_{u,i} \log \frac{r_{u,i}}{w_u^T h_i} - r_{u,i} + w_u^T h_i.$$

The prize of one million dollars was offered by the Netflix for a team which would improve the movie recommendations for users. The one of used methods was the matrix factorization. The final victory was brought by different complex models which are part of several improvements to the main matrix factorization model. Some of them are listed below:

1. Checking additional specific users and specific services to clarify systematic errors in ratings, for example, taking advantages of the fact that popular videos are highly appreciated on the average.
2. Using the temporary dynamic of rating

$$J(W, H) = \sum_{(i,u) \in L} \left( r_{u,i}(t) - b_u(t) - \hat{r} - w_u^T(t) h_i \right)^2,$$

where  $t$  designates counting of time, and  $W$  includes time-dependent dimension of user space.

In many cases, like ratings from 1 to 5 “very bad – very good”, only implicit preferences are available. For example, in such data the records of transactions containing information on products purchased by clients or navigation of the client on the seller’s website are collected. This information can be used for forming negative examples that will allow not to offer clients any service or goods which they do not need. It is difficult

to collect and support such information, taking into account quickly changing business environment. The method of matrix factorization can be used for the solution of such problems if the entering weight  $c_{u,i}$  characterizing trust degree can accept negative values

$$J(W, H) = \sum_{(u,i) \in L} c_{u,i} (r_{u,i} - w_u^T h_i)^2.$$

**Mix of the hidden profiles.** Let  $Y = \{y_m\}_{m=1}^M$  be a set of items and  $U = \{u_n\}_{n=1}^N$  be a great number of the registered users. The rating matrix  $R$  has  $N \times M$  dimensions. The rating established by the user  $u_n$  for item  $y_m$  is designated  $r_{n,m}$ . Observed ratings are kept in the list  $D = \{(u, y, r)_i\}_{i=1}^L$ . Let's notice that not all elements have ratings from all user. Let's describe a probabilistic mixed model of the hidden profiles. The idea is present an assessment of user behavior in a simple generation model. Let's consider two versions of probabilistic approach. The first assumes that there are user groups with essentially the same behavioral rating. The second model allows each user to have an own rating profile, but this profile is presented in the mixed form of several typical profiles of ratings. Let's notice that the rating of a profile is defined by distribution of ratings on a full range of items, on the basis  $P(r, y | \theta)$ , where  $r = (r_1, r_2, \dots, r_M)^T$  and  $y = (y_1, y_2, \dots, y_M)^T$  with value  $y_m = 1$  if the element  $m$  received the rating ( $r_m \neq \emptyset$ ) and  $y_m = 0$  in the case when the item  $m$  did not receive the rating ( $r_m = \emptyset$ ). When forecasting rating it is usually known on what element it is necessary to do the forecast and thus it is enough to define conditional probability of rating the profile  $P(r | y; \theta)$ , considering that distribution is parametrized by  $\theta$  (see more in Lim, Teh 2007).

In the beginning we will consider the rating constructed on binary information – whether there was an interaction between the user  $u_n$  and element  $y_m$  or not. In this case the rating of a profile is defined through  $P(y | \theta)$  the probability of receiving (according to the available information) rating for each element. In this case, using Bernoulli's distribution, we receive

$$P(y | \theta) = \prod_m \beta_m^{y_m} (1 - \beta_m)^{1 - y_m}.$$

Set of parameters  $\theta \equiv (\beta_1, \dots, \beta_M)^T$  and  $0 \leq \beta_m \leq 1$  defines probability of interaction between the user and the  $y_m$ .

An alternative to Bernoulli distribution is multinomial distribution. In this case there is an additional restriction between parameters:  $\theta \equiv \beta = (\beta_1, \dots, \beta_M)^T$  under

a condition  $\sum_m \beta_m = 1$  and  $\beta_m \geq 0$ . Then the probability of the profile can be defined as follows

$$P(y | \theta) \propto Mn(y | \beta, V) = V! \prod_m \beta_m^{y_m} = V! \prod_{y_m=1} \beta_m,$$

where  $V = \sum_m y_m$  is a number of elements having the rating,  $Mn$  is a multinomial distribution. It is interesting that, for multinomial distribution, the unrated elements do not make a direct contribution to probability. It makes the process of probability calculation much more quicker and more effective on big rating matrices. The proportionality mark  $\mu$  shows that the following restrictions  $y_m \in \{0, 1\}$  should be taken into account when carrying out normalization. Let's notice that this factor does not influence the optimization of parameters.

In more general case the multinomial distribution is suitable for the ratings represented by integer numbers  $r \in \{0, 1, 2, \dots\}$ . As in the previous case, we have

$$P(r | \theta) = Mn \left( r | \beta, \sum_m r_m \right).$$

The vector which has not been rated is absent in this expression as the information on the absent ratings is already coded in  $r$ .

The principle of determining the profile rating remains the same for explicit ratings i.e. when ratings are coded taking into account users reviews. Explicit ratings can accept discrete values  $r \in \{1, 2, \dots, r_{\max}\}$  or continuous values  $r \in [r_{\min}, r_{\max}]$  in an ordered set. We can associate the absent values of rating with  $r = \emptyset$ . At first let's consider a simple case when only conditional distribution of rating is required  $P(r | y; \theta)$ . It would be possible to define a profile, receiving ratings, for example, from Gaussian distributions. So, using the parametrization  $\theta \equiv \{(\mu_m, \sigma_m)\}_{m=1}^M$  we receive the probability of a profile as follows

$$P(r | y; \theta) = \prod_{y_m=1} N(r_m | \mu_m, \sigma_m),$$

where the product is calculated only on the basis of a set of observed ratings. In practice, ratings are always forced to accept values from a limited interval, and, then it makes sense to use limited distributions such as the theta distribution, which has two parameters. Besides, if ratings can be chosen from discrete set, binomial distribution or multinomial distribution would be more natural than Gaussian distribution.

Now, taking into account the scheme of forecasting, i.e. information  $P(r | y; \theta)$ , it is possible to combine implicit and explicit ratings in a profile. Standard approach is the combination of a multinomial distribution for the missing information and Gaussian distributions for the available values of rating. Therefore, we have  $\theta \equiv \{(\beta_m, \mu_m, \sigma_m)\}$  and probability of a profile as follows

$$P(r | y; \theta) = P(r | y; \{(\mu_m, \sigma_m)\}) P(y | \beta) \propto \prod_{y_m=1} \beta_m N(r_m | \mu_m, \sigma_m).$$

Let's notice that if the purpose is only to predict the rating couples user item, i.e. to determine a good evaluation of  $P(r|u, y)$ , then taking the lack of ratings in model into account will not probably be an advantage.

**The clustering hidden profile.** The idea of the clustering hidden profile is that each user is given one rating profile from the set of typical profiles. Let's assume that there are  $K$  models of rating profiles. Each profile has its own parametrization, therefore, the full range of profiles parameters is  $\theta \equiv \{\theta_k\}_{k=1}^K$ . The user  $u$  is given a vector of indexes  $z = [z_1, \dots, z_K]$  and  $z_k = 1$  if the user corresponds to  $k$ -th profile and  $z_k = 0$  otherwise. Then the probability that the user corresponds to  $k$ -th profile can be written as follows

$$P(r, y | u; z, \theta) = \prod_k P(r, y | \theta_k)^{z_k}.$$

### 10.1.2. The content-oriented recommendations

Collaborative filtering recommendations use only user ratings that are not likely to obtain any forecasts without specifying of certain users or items. However, it is possible to make more personalized recommendations, knowing more about the user, for example, having demographic customer information or information about kind of films the client prefers. The content based (CB) recommendations belong to the approaches that predict a customer behavior by analyzing content referring to the item (service, element) description taking into account the interests of users.

The scheme below (see Fig. 10.4) reflects the general principles of content filtering.

Numerous studies in this area are concentrated on finding the recommendations for goods  $y$  or services with a relevant text content, for example, on the web pages of goods or services containing users descriptions and responses (see more in Kuppussamy, Aghila 2011, Marlin 2004, Sarwar et al. 2000, 2001). Thus, it is possible to treat this problem, as a problem of information search. It is possible to treat the task connected with the user preferences as a query, and to correlate the rating of the text with the



relevance of the request (for more details see in Yang 1999, Yang, Pedersen 1997, Yang, Xin Liu 1999). Then the documents in each rating category are given the vectors of TF-IDF (term frequency, inverse document frequency – see in section 6.2 devoted to qualifiers). Then averaging is carried out which allows a vector prototype to be received for each category of users. To classify a new document the corresponding vector is compared with a prototype of each category. The prediction of rating is based on a cosine value of the angle between these vectors.

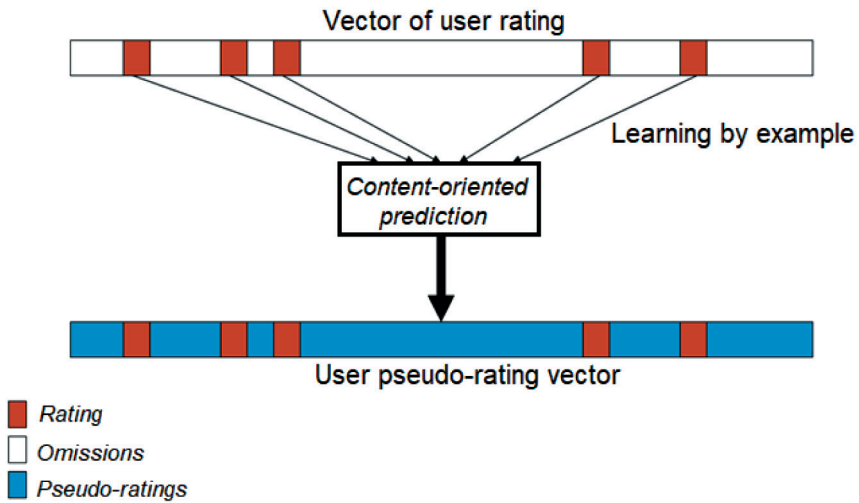


Fig. 10.4. The general scheme of content-filtering

An alternative to the approaches based on information search is using of stochastic models, for example, naive Bayesian classification, creation of a decision tree, neural networks and so forth.

### 10.1.3. Profiles of users

The profile of user interests is used in the majority of recommendatory systems. This profile can consist of different information types. We will concentrate on two information types:

1. Model of the user settings, i.e. the description of elements types representing interests of users. There are many possible alternative suggestions for this description, but the general rule is the availability of a function which for any element predicts probability as far as the user is interested in this item (service). To make recommendations more effective, this function can be used for receiving  $n$  elements which are most likely of interest to the user.

2. Stories of the user interaction with system recommendations. It can include storage of elements which the user looked through together with other information about the client actions (for example, that the user purchased the items specified by the other client). History of the requests entered by the user can be one more information.

In the user settings, the system of the recommendation provides the interface which allows users to construct an idea of his interests. For this purpose tags are often used that allows the user to choose one or another known value of attributes, for example, to choose favorite sections of news, or a movie genre. In other cases, the HTML form allows the user to enter words, for example, a name of the musician or author which are interesting to the user. As soon as the user enter this information into the database, the process of search uses them for finding elements which answer the criteria determined for the client and displays them to the user. At the same time there are several restrictions of the system in the user settings. First, they demand some effort from the user, and it is impossible to expect a large number of users who will do this work. Secondly, settings of the system do not provide a method of determination of a sequence (rating) of elements.

The recommendation system based on history of the user actions should have the rules of recommending other products. For example, the system may contain the rule which recommends continuation of a book or a movie to clients who have already purchased the book or the movie from this series. Other rule can recommend a new compact disk for users who purchased disks of this contractor earlier. In some situations it is reasonable to recommend goods to the user if he purchased it earlier and in other situations, it is not. For instance, the system should offer fixed elements that are easily worn such as edges of the razor or a cartridge, and additionally at the same time recommend a mobile phone or Webcam.

#### **10.1.4. Training a user model**

Creating the behavior model of the client based on the user history is one of forms of the recommendation systems education. Let's consider classification of training algorithms. Such algorithms are a key component of creating recommendations systems. As it was already noted above, the algorithm of training includes function which provides assessment of probability that the user will pay for goods (items, service). This probability can be used for sorting the list of recommendations. Traditional algorithms of machine training are intended for work on structured data. When working with documents, the text at first are transformed into structured data, using a small subset of terms as attributes. Let's look at the overview of several most popular algorithms.

## 1. Decision tree

The decision tree uses recursively sectioned data on which it is trained. In case of using texts the documents are divided consistently into subgroups before formation of such subgroups that contain only one copies class. As a rule, as selection criterion an availability or a lack of the characteristic word or phrase is used. Decision trees are well studied when using structured data (see more in section 6.4). Enriching knowledge on decision tree can be found for example in (Ghazanfar, Prügel-Bennet 2010 Kohavi 1996, *Naive Bayes...*, Baldwin, Xie 2005, Olaru, Wehenkel 2003, Rokach, Maimon 2014, Smith 2002).

## 2. Behavior of the nearest neighbor

The algorithm of the nearest neighbor just stores all stages and levels of data training. To classify the new, not marked point, the algorithm compares them to all kept elements and, using similarity, defines “the nearest neighbor” and consolidates recommendations with actions of the nearest neighbors. Usage of an algorithm of the nearest neighbor depends on a data type. Quite often this algorithm is used for structured data with the Euclidean metric, or for vector model with the similarity criterion given by the cosine of the angle between the vectors. The effectiveness of this algorithm for comparing texts referring to one topic is sufficient. This approach is often used for personalization of news. It might be much recommendable to dive in (Ricci et al. 2011, Pedregosa et al. 2011, Perkins 2014, Richert, Coelho 2013, Sammut, Webb 2017).

## 3. Relevant compliance and Rocchio algorithm

As with the vector space model, success in document search depends on user’s ability to create requests, select a set of keywords, as well as methods that help users determine progressive requests based on previous search results. The right document search is the focus of many studies. These method are usually called the feedback (the relevant compliance). The Rocchio algorithm (Rocchio 1971) is a widely used feedback algorithm. The algorithm is based on modification of the request through the weighed prototypes of the relevant and not relevant documents. This approach creates two prototypes of the document. The first prototype relates to all corresponding documents and the second relies on non-compliance with the documents. This is formalized as follows

$$Q_{i+1} = \alpha Q_i + \beta \sum_{\text{rel}} \frac{D_i}{|D_i|} - \gamma \sum_{\text{nonrel}} \frac{D_i}{|D_i|}.$$

Here  $Q_i$  is a user request for iterations of  $i$  and  $\alpha, \beta, \gamma$  are parameters which control influence of the initial request and two prototypes on change in result of the request. The main idea is to gradually move the request vector towards clusters of relevant documents and far from irrelevant documents.

#### 4. Linear qualifiers

As it was mentioned above (see the section 6.5 devoted discriminant analysis) the algorithms, based on the linear borders i.e. the hyperplanes dividing sets of decisions in multidimensional space are called linear qualifiers. There is a large number of algorithms which get to this category, and many of them are successfully applied in the task of text classification. The reader can go into greater details in for instance (Backer, Mc Callum 1998, Pedregosa et al. 2011, Sammut, Webb 2017).

#### 5. Probabilistic methods and naive Bayesian qualifier

Unlike the vector model, which does not have a sufficient theoretical justification, probabilistic approaches of classification in the basis have a good theoretical explanation. At the moment the naive Bayesian classifier (see the section 6.4 devoted to classifier) is recognized as exceptionally well working algorithm of text classification.

So, we need to find the most probable hypothesis  $h \in H$  on the condition of availability of data  $D$ , meaning that it is necessary to maximize  $P(h|D)$ . To achieve it we can use the Bayes theorem (6.1)

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

i.e. it is required to find  $h = \arg \max_{h \in H} P(h|D) = \arg \max_{h \in H} P(D|h)P(h)$ , and if hypotheses are equiprobable, then  $h = \arg \max_{h \in H} P(D|h)$ .

Let's compare different options of the simplified Bayes algorithm – multinomial model and Bernoulli's model. Both models are based on the assumption that textual records generate the probability model described by mixture

$$P(d_i | \theta) = \sum_{j=1}^{|C|} P(c_j | \theta) P(d_i | c_j; \theta).$$

Here, to each class  $c_j$  there corresponds the mix component which will be parametrized by subsets not crossed with  $\theta$ . As soon as the parameters  $\hat{\theta}$  from the training data are received, the probability of a posteriori belonging of the test text document to

a class can be defined using Bayes' theorem as follows

$$P(c_j | d_i; \hat{\theta}) = \frac{P(c_j | \hat{\theta})P(d_i | c_j; \hat{\theta})}{P(d_i | \hat{\theta})}.$$

The multidimensional Bernoulli model and multinomial model differ in a method of determining the value  $P(d_i | c_j; \theta)$ .

Let  $V = \{w_t\}_{t=1}^{|V|}$  be a dictionary. Then the document  $d_i$  is a vector of length  $|V|$ ,  $B_i$  consist of bits,  $B_{i,t} = 1$  if the word  $w_t$  occurs in the document  $d_i$ , otherwise  $B_{i,t} = 0$ .

The multidimensional formulation of Bernoulli distribution for problems referring to the text classification assumes that each document is presented in the form of a binary vector over the space of all words from the dictionary  $V$ . For each element  $B_{i,t}$  this vector specifies whether the word appears in the document at least once. Following naive Bayes theory, the probability of each word entering the document does not depend on other words. Consequently it is possible to present it in the following form

$$P(d_i | c_j; \theta) = \prod_{t=1}^{|V|} (B_{i,t} P(w_t | c_j; \theta) + (1 - B_{i,t})(1 - P(w_t | c_j; \theta))).$$

For training of such classifier it is necessary to calculate probabilities  $P(w_t | c_j; \theta)$ . Let the following elements be given: a set of documents  $D = \{d_i\}_{i=1}^{|D|}$  which are already placed within the class  $c_j$ , the dictionary  $V = \{w_t\}_{t=1}^{|V|}$  and bits  $B_{i,t}$  (we know documents).

Then you can calculate the optimal probabilities of estimates  $P(w_t | c_j; \theta)$  taking into account that one or the other word meets in this or another class (Laplacian smoothing)

$$P(w_t | c_j; \theta) = \frac{1 + \sum_{i=1}^{|D|} B_{i,t} P(c_j | d_i)}{2 + \sum_{i=1}^{|D|} P(c_j | d_i)}.$$

The prior probabilities of classes can be calculated as follows

$$P(c_j) = \frac{1}{|D|} \sum_{i=1}^{|D|} P(c_j | d_i).$$

And classification can take such a form

$$\begin{aligned}
 c &= \arg \max_j P(c_j) P(d_i | c_j; \theta) = \\
 &= \arg \max_j \left( \frac{1}{|D|} \sum_{i=1}^{|D|} P(c_j | d_i) \right) \prod_{t=1}^{|V|} \left( B_{i,t} P(w_t | c_j; \theta) + (1 - B_{i,t}) (1 - P(w_t | c_j; \theta)) \right) \\
 &= \arg \max_j \left( \log \left( \sum_{i=1}^{|D|} P(c_j | d_i) \right) + \sum_{t=1}^{|V|} \log \left( B_{i,t} P(w_t | c_j; \theta) + (1 - B_{i,t}) (1 - P(w_t | c_j; \theta)) \right) \right)
 \end{aligned}$$

Unlike binary submission of the document multidimensional model of Bernoulli, the multinomial formulation considers the frequency of word loss. This model assumes that documents are generated by the sequence of independent tests with multinomial distribution of probabilities. Let there be given a set of documents  $D = \{d_i\}_{i=1}^{|D|}$ , which are already distributed within the class  $c_j$ , the dictionary  $V = \{w_t\}_{t=1}^{|V|}$  and  $N_{i,t}$  the number of word occurrence  $w_t$  in the document  $d_i$ . Besides, naive Bayes allows to define  $P(d_i | c_j; \theta)$  based on probabilities of a meeting separate words

$$P(d_i | c_j; \theta) = P(|d_i|) \prod_{t=1}^{|d_i|} P(w_t | c_j; \theta)^{N_{i,t}}.$$

At the same time, the probability  $P(w_t | c_j; \theta)$  can be estimated as follows

$$P(w_t | c_j; \theta) = \frac{1 + \sum_{i=1}^{|D|} B_{i,t} P(c_j | d_i)}{|V| + \sum_{s=1}^{|V|} \sum_{i=1}^{|D|} N_{i,s} P(c_j | d_i)}.$$

The prior probabilities of classes can be calculate in the following way

$$P(c_j) = \frac{1}{|D|} \sum_{i=1}^{|D|} P(c_j | d_i).$$

Then the classification takes the following form

$$\begin{aligned}
 c &= \arg \max_j P(c_j)P(d_i | c_j; \theta) = \\
 &= \arg \max_j \left( \frac{1}{|D|} \sum_{i=1}^{|D|} P(c_j | d_i) \right) P(|d_i|) |d_i|! \prod_{t=1}^{|V|} \frac{1}{N_{i,t}!} P(w_t | c_j; \theta)^{N_{i,t}} \\
 &= \arg \max_j \left( \log \left( \sum_{i=1}^{|D|} P(c_j | d_i) \right) + \sum_{t=1}^{|V|} N_{i,t} \log P(w_t | c_j; \theta) \right).
 \end{aligned}$$

Experience shows that the simplified Bayes algorithm with multinomial distribution is more effective than this algorithm with multidimensional Bernoulli distribution.

The reader is welcome to dive in (Ghazanfar, Prügel-Benet 2010, *Naive Bayes...*, Sebastiani 2002, Ricci et al. 2011, Pedregosa et al. 2011).

### 10.1.5. Hybrid approaches

To use strengths of both approaches and to receive more effective recommendation system, it is necessary to use hybrid approaches (see for example (Ghazanfar, Prügel-Benet 2010)). One naive approach consists that on the basis of the content and joint methods of filtering separate lists of recommendations are formed, and then those lists are combined to receive the final weighted consolidation of two forecasts where weight increases with the number of the users choosing the element.

The interesting idea is transformation of disperse matrices of user ratings to the filled matrices, and then, using the CF methods, obtaining recommendations. In particular, it is possible to use the naive Bayesian qualifier for training on the documents describing the rating of each user and for filling the missing ratings. As a result we receive a matrix of pseudo-ratings, using which, we find the neighbors similar to the active user, and we receive forecasts by means of Pearson's correlation. There are other hybrid approaches based on joint filtering, but also supporting a profile of each user.

The quality of the recommendation system can be estimated by comparison of the recommendation for a test set of the known ratings of users (see Herlocker et al. 2004). The most often used metrics is the mean absolute error (Mean Absolute Error – MAE).

It is defined as the average absolute difference between predicted and actual ratings and can take the following form

$$\text{MAE} = \frac{\sum_{\{u,i\}} |p_{u,i} - r_{u,i}|}{N},$$

where  $p_{u,i}$  is the predicted rating of an element  $i$  by the user  $u$  on,  $r_{u,i}$  is the actual rating, and  $N$  is the total quantity of the ratings received during testing.

Another, traditionally used metric, is root mean square deviation (RMSE-Root Mean Squared Error) which places the bigger emphasis on big absolute error, and can be determined by the following formula

$$\text{RMSE} = \sqrt{\frac{\sum_{\{u,i\}} (p_{u,i} - r_{u,i})^2}{N}}.$$

Usage of these metrics leads to considering all elements equally. However, for the majority of the recommendation systems, the main task is to predict the user behavior as accurately as possible. In the context of information search systems (ISS – Information Search System), the identified item against the bad elements in the background can be considered as distinction between the “corresponding” and “important” elements. Here other criteria can be used as a test of accuracy, for example, a  $F$ -measure of the balanced mistake,  $\tau$  – Kendall’s correlation and so on.

Let’s note several problems.

Innovations and new users create a serious problem for the recommendation system. In total these problems are called a problem of “cold start”. The first of them arises in the system of joint filtering when service cannot be recommended until a user estimates it. It concerns not only innovations, but also the existing services that are especially harmful for users with special (exotic) tastes. In general it is difficult to solve a problem of new users because without any information about the user preferences it is not possible to find similar users or to construct his profile.

Other problem is the fraud. As soon as the recommending systems of commercial websites began to play a significant role in impact on profitability. It led to participation of unfair suppliers in different forms of fraud with the purpose to get some advantage of using the recommendation systems. As a rule, they try to inflate popularity of own products (push attack – increase the prediction value of a target element) or to downgrade the competitors (nuke attack – decrease the prediction value of a target element). Such attacks usually provide creation of dummy profiles, and use information on system operation.



The good example of using mentioned method is the work by (Ghazanfar, Prügel-Benet 2010).

## 10.2. Analysis of client environments

**The client environment** is a set of clients (users, subjects) who are regularly using the fixed set of services (goods, resources, objects, items, services). It is supposed that actions of clients are recorded in electronic form. Examples of actions can be: using service or purchasing goods, estimating (rating) service or goods, making requests for information, fee, the choice of a tariff plan, participation in a marketing event, receiving a bonus from the company, refusing service, etc.

**The analysis of client environments** (Customer Environment Analysis – CEA) is the technology of processing client action protocols allowing to calculate effectively mutually approved estimating similarity of clients and services. It can be used for solving such business challenges as automating marketing research, forming the directed offers to clients, personalizing services, increasing satisfaction and customer loyalty, more effective attraction and customer retention.

The CEA technology can be used for creating the recommendation systems, personalization of offers (targeting, direct marketing) and creating customer relationship management systems (Customer Relationship Management – CRM).

Directions, closest to CEA, are collaborative filtering (Collaborative Filtering – CF) and the correspondence analysis (Correspondence Analysis – CA). CEA has two main differences relating to CF:

- CEA is aimed at receiving mutually approved estimating similarity of clients and services. Clients and services are considered as equal, dual entities. Any analysis made concerning clients can be reflected in services and vice versa. Methods of collaborative filtering, especially simple, do not allow such duality.
- CEA considers all complex of the tasks and methods connected with further use of the received similarity estimates for visualizing, clustering, classifying and forecasting of customer behavior and finally with solving the listed above business challenges. Works on collaborative filtering are in most cases limited to narrow problem definitions like predicting ratings or forming recommendations.

Below we present some examples of using the CEA.

### 10.2.1. Examples of client environments

Client environments arise in the most different spheres of business, and not only business. It is possible to speak about client circles of item producers, dealer networks,

networks of supermarkets, telecom operators, plastic card issuers, libraries, online stores, search engines, social networks, forums, blogs etc.

Also such appendices CEA in which the terms “clients” and “services” are hardly applicable, for example the analysis of texts or the analysis of parliamentary election results are possible. However mathematical methods of data handling remain the same.

### **10.2.2. Retail chain stores**

Let’s denote that “services” are goods (items), “clients” are the regular customers having the discount card and “actions of clients” are purchases of goods (items).

It can be listed some examples of tasks:

- Make the client a directed proposal of those goods (items) which with a high probability will be pleasant to him. The personal offer can be printed on a reverse side of the bill or be moved on the special terminal at the request of the client.
- Just in time let to the client know where there are new goods (items) about which still very few people know, but which with a high probability will interest this client.

### **10.2.3. Mobile operators**

Let’s assume that “services” are different services (types of connections), “clients” are subscribers of network and “actions of clients” are the calls of different types (entering, outgoing, long-distance, international, the SMS, MMS, etc.), but also payments, connections and shutdowns of services, changes of tariff plans, appeals to the service center etc.

In this case there can be some examples of such tasks:

- Forecasting clients resignation (churn prediction), on the basis of similarity to already quitting clients.
- Segmentation of client base and allocation of target client groups.
- Identification of similar services when forming package offers.
- Detection of unusual or potentially dangerous customer behavior (fraud detection).

### **10.2.4. Online stores of books, audio and video of other products**

The assumption in this case are as follows: “services” are goods (items – books, disks, movies, etc.), “clients” are regular customers and actions of clients are either purchases of goods (items), or assessments (ratings) of goods.

Examples of tasks can take the following form:

- Predict the ratings of goods (items) for the user and offer the customer the list of the most interesting goods (items).
- Offer a personal discount for joint purchase of several goods (items) (cross-selling).
- Just in time inform the client about new interesting to the customer goods (item) (up-selling).

### **10.2.5. Search engines**

Let's define that "services" are the pages or documents offered as search results, "clients" are users of the search engine. "Actions of clients" are transitions from the page of search results to the found document. In this appendix the CEA technology adjoins the analysis of a web (web mining), more precisely, to the analysis of the web user behavior (web usage mining).

Examples of tasks:

- Range search results in such order that the documents high on the list are the ones with high probability interesting to the user.
- Place targeted advertising on the page, suggesting the user to visit the websites, with a high probability interesting to the customer, exactly at present.
- Find for the website the list of the closest websites (for example, for automatic generating the page of useful links).
- Find for the website a list of the websites, the closest to the searched one (for automatic generating the personalized list of the recommended references).

### **10.2.6. Parliamentary elections**

Here as "services" we define political parties acts, "clients" are territorial subjects of the federation, territorial constituencies or polling precincts. "Actions of the client" are the votes given to the parties.

Tasks are connected generally with interpretation of election results:

- Make a ranking of political parties based on similarity concerning any set party.
- Make a ranking of regions based on similarity concerning any set region.
- Understand and visualize (for example, by means of the card of similarity) a political range of parties.
- Allocate votes to similar political parties.
- Allocate votes to regions in which this political party could draw votes for other parties.

### **10.2.7. Analysis of texts**

In this case “services” are keywords or expressions, “clients” are texts. “Action of the client” corresponds to the fact that this keyword occurs in this text.

Examples of tasks:

- Automatic classifying and clustering large volumes of textual records or news flows.
- Document retrieval based on the similarity to the document.
- Search of the most complete and relevant documents on a specific subject.

### **10.2.8. Social networks**

In the elementary case, “services” are pages (blog entries, personal pages of users, sections of a forum), “clients” are users of social service. Actions of the client are: visiting the webpage, viewing messages, creating own messages, adding/removing friends, etc. Social networks are more difficult example of the customer environment analysis as it is necessary to apply the analysis of text information. Generally there are not two types of the interconnected entities (clients and services), or three types: users, pages and keywords.

Examples of tasks:

- Personal offer of webpages, forums, contacts, interesting to the user.
- The automatic personalized classifying and clustering webpages, forums, contacts.
- Search of adherents (like-minded people), similar people (neighbors).

# Appendix

## Basic information

### A.1. Background information on linear algebra

Improving basic knowledge on linear algebra and probability theory will be much-needed for having a better grasp of next sections. Without getting into details, in this paragraph we will provide necessary information relating to the subsequent sections.

Let's remind that for the matrix  $A$  the  $n \times m$  size we understand replacement of rows with columns with the same numbers as transposing, in particular, if

$$X = \begin{bmatrix} x_{1,1} & x_{1,2} \\ x_{2,1} & x_{2,2} \\ x_{3,1} & x_{3,2} \\ x_{4,1} & x_{4,2} \end{bmatrix},$$

that

$$X^T = \begin{bmatrix} x_{1,1} & x_{2,1} & x_{3,1} & x_{4,1} \\ x_{1,2} & x_{2,2} & x_{3,2} & x_{4,2} \end{bmatrix}.$$

The important characteristic used further is the scalar product (dot product) of two vectors

$$\langle X, Y \rangle = X^T Y = x_1 y_1 + x_2 y_2 + \dots + x_n y_n = \sum_{i=1}^n x_i y_i,$$

where

$$X^T = [x_1 \quad x_2 \quad \dots \quad x_n] \text{ and } Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}.$$

On the other hand, noticing that  $\langle X, Y \rangle = |X||Y|\cos\theta$ , where  $\theta$  is the angle between vectors  $X$  and  $Y$ , we derive  $\cos\theta$  as

$$\cos\theta = \frac{X^T Y}{|X||Y|}.$$

Thus, the dot product (scalar) characterizes the deviation of the vector  $X$  from the vector  $Y$ . The Euclidean metrics or length of the vector is defined as the number

$$|X| = \sqrt{\langle X, X \rangle} = \sqrt{\sum_{i=1}^n x_i^2},$$

and Euclidean distance between two vectors we will call number

$$|X - Y| = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}.$$

Vectors  $X_1, X_2, \dots, X_m$  are called linearly independent if equality

$$\alpha_1 X_1 + \alpha_2 X_2 + \dots + \alpha_m X_m = 0,$$

then and only then when all  $\alpha_i = 0, i = 1, 2, \dots, m$ .

If this condition is satisfied but at least at one  $\alpha_i \neq 0$ , then the system of vectors is linearly dependent.

The set of all  $m$ -dimension linearly independent vectors generate the vector space  $V$  of the same dimension.

The set of vectors  $\{U_1, U_2, \dots, U_m\}$  is called a basis of the vector space  $V$  if for  $\forall v \in V$  there will be such set  $\{\alpha_i\}_{i=1}^m$ , that

$$v = \alpha_1 U_1 + \alpha_2 U_2 + \dots + \alpha_m U_m.$$

Basis  $\{U_1, U_2, \dots, U_m\}$  is orthogonal if  $U_i \perp U_j$  for  $\forall i \neq j$  (i.e.  $\langle U_i, U_j \rangle = 0$ ) and if at the same time  $|U_i| = 1, i = 1, \dots, m$ , the basis is orthonormal.

If for the square matrix  $A$  the size  $m \times m$  there will exist the nonzero vector of the  $X$  such that the condition is satisfied  $AX = \lambda X$ , that  $X$  is the eigenvector of the matrix  $A$  and the number  $\lambda$  is called the matrix eigenvalue. Thus, the linear transformation realized by the matrix  $A$ , transfers the eigenvector of  $X$  in colinear, sent to the same party if  $\lambda > 0$ , and to the return if  $\lambda < 0$ .

Let's note several important properties of eigenvalues.

- If the matrix  $A$  is valid and symmetric (that is  $A^T = A$ ), all eigenvalues are valid.
- If the matrix  $A$  is not singular (that is its rank is equal to number of rows), then its eigenvalues will be not zero.
- If the matrix  $A$  is positively defined (that is  $X^TAX > 0$ ), all its eigenvalues are positive.

In order to have more detailed understanding of using linear algebra, the reader should get familiarized within (Press et al. 2007, Rao, Toutenburg 1999).

## A.2. Background information on probability theory

Let  $\Omega$  be the set of all possible outcomes of some events, and  $S$  – event algebra, set of subsets of the set  $\Omega$ , for which the following conditions are satisfied:

1.  $S$  includes impossible and reliable events.
2. If events  $A_1, A_2, \dots$  (the finite or countable set) belongs to  $S$ , that  $S$  possesses the union, the intersection and the subtraction of these events.

Probability function  $P(A)$  is defined on  $S$ , accepting real values from zero to one and satisfying axioms:

- Nonnegativity axiom:  $\forall A \in S; P(A) \geq 0$ .
- Normalization axiom: the probability of the certain event is equal to unit:  $P(\Omega) = 1$ .
- Additivity axiom: the probability of the union of disjoint events is equal to the sum of probabilities of these events: if  $A_i \cap A_j = \emptyset (i \neq j)$ , that

$$P\left(\bigcup_k A_k\right) = \sum_k P(A_k).$$

Let's give the basis properties of probability.

1.  $P(\emptyset) = 0$ ;
2.  $P(A) \leq 1$ ;
3.  $A \subset B \Rightarrow P(A) < P(B)$ ;
4.  $P(A \cup B) = P(A) + P(B) - P(A \cap B)$ .

The probability of the event  $A$  provided that there was given the event  $B$ , is called conditional probability  $P(A|B)$ . It is calculated as follows

$$P(A|B) = \frac{P(A \cap B)}{P(B)}.$$

If events  $A$  and  $B$  are independent, then  $P(A \cap B) = P(A)P(B)$ , and for conditional probability it is possible to obtain

$$P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{P(A)P(B)}{P(B)} = P(A).$$

Let

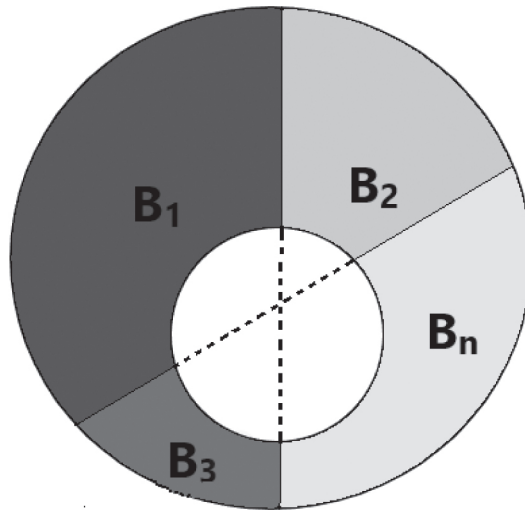
$$A = (A \cap B_1) \cup (A \cap B_2) \cup (A \cap B_3) \cup \dots \cup (A \cap B_n),$$

then from the formula of conditional probability we receive the following equation (A.1) (see Fig. A.1)

$$P(A) = P(A|B_1)P(B_1) + P(A|B_2)P(B_2) + \dots + P(A|B_n)P(B_n) \quad (\text{A.1})$$

that is

$$P(A) = \sum_{k=1}^n P(A|B_k)P(B_k).$$



**Fig. A.1.** Illustration of the composite probability formula

The important role in further reasonings is played by Bayes formula or the theorem of hypotheses. This statement allows to reevaluate the probability of hypotheses  $B_i$ , accepted before experience (event) and called by the priori (a priori – before



experience) by results of already made experiment, that is using a posteriori (a posteriori – after experience) probabilities. If  $B_1, B_2, \dots, B_n$  are the making sets of  $S$ , then probability that the event  $A$  will lead to the event  $B_i$  is equal

$$P(B_i | A) = \frac{P(B_i \cap A)}{P(A)} = \frac{P(A | B_i)P(B_i)}{\sum_{k=1}^n P(A | B_k)P(B_k)}.$$

Random variable (random quantity)  $X$  is called the function defined on the set of events  $\Omega$ , which relates each simple event  $\omega$  to the corresponding number  $X(\omega)$ . The random variable can be discrete and continuous.

Any rule allowing to find probabilities of any events  $A \subseteq S$ , is called the distribution function of the random variable, and at the same time say that the random variable submits to this distribution.

Distribution function of the random variable  $X$  is a function  $F(x)$ , which relates for any  $x \in R$  to the probability of the event  $\{X \leq x\}$

$$F(x) = P(\{X \leq x\}).$$

Distribution function has the following properties

1.  $0 \leq F(x) \leq 1$ .
2.  $F(x)$  is non-decreasing function, i.e. if  $x_2 \leq x_1$  then  $F(x_1) \leq F(x_2)$ .
3.  $F(-\infty) = 0, \quad F(+\infty) = 1$ .
4.  $P(\{a < X \leq b\}) = F(b) - F(a)$ .

For the discrete random variable  $X$  let

$$p(x) = P(X = x),$$

then

$$F(x) = P(X \leq x) = \sum_{a \leq x} P(X = a) = \sum_{a \leq x} p(a).$$

For the continuous random variable the function  $f(x) \geq 0$  is called probability density function if

$$F(a) = P(X \leq a) = \int_{-\infty}^a f(x) dx.$$

Hence

$$P(a < x \leq b) = \int_a^b f(x) dx.$$

Furthermore

$$\frac{d}{dx} F(x) = f(x),$$

besides

$$P(x = a) = \int_a^a f(x) dx = 0 \text{ and } P(-\infty \leq x \leq \infty) = \int_{-\infty}^{\infty} f(x) dx = 1.$$

Let's consider some important characteristics of the random variable. The first moment of the random variable is called the average value. For the discrete case the average has the appearance

$$\mu = E(X) = \sum_x xp(x),$$

for the continuous case

$$\mu = E(X) = \int_{-\infty}^{\infty} xf(x) dx.$$

The quantity

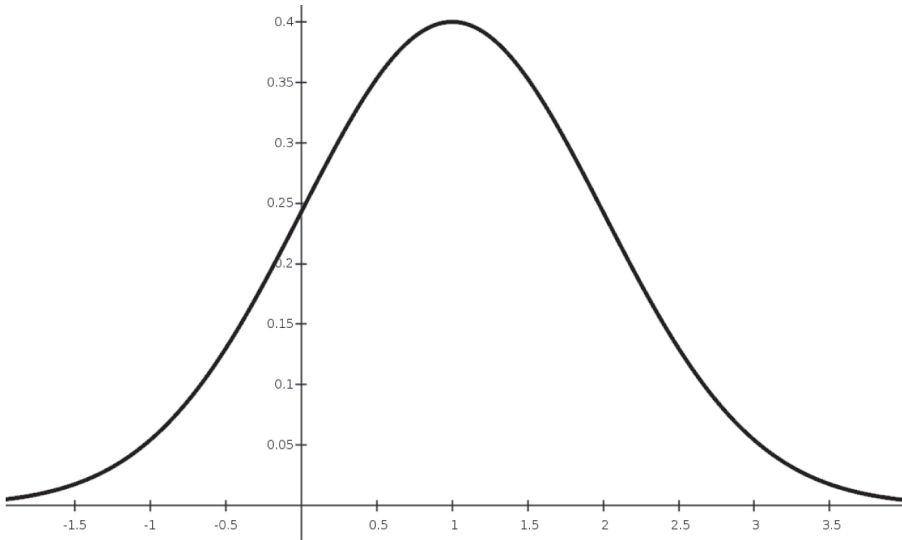
$$\sigma^2 = \text{var}(X) = E\left(X - E(X)\right)^2$$

is called the variation or dispersion. This number characterizes dispersion of the random variable, and  $\sigma$  is called root-mean-square deviation of the random variable from the ensemble average.

The special role in probability theory is played by the normal distribution (Gauss's law) that is caused, first and foremost, by the fact that there is a limit law to which other distribution laws come down (under certain conditions).

Let's say that the continuous random variable of  $X$  is distributed under the normal law  $N(\mu, \sigma)$ , then its density function has the following appearance (Gauss's function) (A.2) and it is illustrated in Figure A.2.

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right), \quad x \in R \quad (\text{A.2})$$



**Fig. A.2.** Function graph of density of normal distributions (Gauss's function) for  $\mu = 1, \sigma = 1$

Let's notice that, according to the normal law the most different sizes, (for example, of the error of measurements, details wear in mechanisms, the weight of fruits and animals, human height, rate fluctuations of actions and many other things) are distributed.

Despite importance of the one-dimensional case, for us, the consideration of the case of many variables is more urgent.

The arranged set  $(X_1, X_2, \dots, X_n)$  of random quantities  $X_i$  ( $i = 1, 2, \dots, n$ ) on the same set  $\Omega$  is called the  $n$ -dimensional random variable. One-dimensional random quantities  $X_1, X_2, \dots, X_n$  are called components of the  $n$ -dimensional random variable. It is convenient to consider components as the coordinate of the accidental vector  $X = (X_1, X_2, \dots, X_n)$  in  $n$ -dimensional space of measurements.

The arranged couple  $(X, Y)$  of two random quantities is called the two-dimensional random variable. Total characteristic of system  $(X, Y)$  is distribution of probabilities specifying area in regard to possible values of random quantities and probability of these values.

Distribution function of the two-dimensional random variable  $(X, Y)$  function is called  $F(x, y)$ , which for any two real numbers  $x$  and  $y$  is equal to probability of joint performance of two events  $\{X \leq x\}$  and  $\{Y \leq y\}$ , that is

$$F(x, y) = P\{X \leq x, Y \leq y\} = P(\omega \in \Omega: X(\omega) \leq x, Y(\omega) \leq y).$$

For discrete accidental couple  $(X, Y)$  as density function we will put

$$p(x, y) = P(X = x, Y = y).$$

In the continuous case the function  $f(x, y) \geq 0$  is called the density function of probability distribution, if

$$F(a, b) = P(X \leq a, Y \leq b) = \int_{-\infty}^a \int_{-\infty}^b f(x, y) dx dy.$$

Then

$$P(a \leq x \leq b, c \leq y \leq d) = \int_a^b \int_c^d f(x, y) dx dy.$$

Let's notice that

$$\frac{\partial^2}{\partial x \partial y} F(x, y) = f(x, y).$$

The important role in further researches is played by the joint moment of the second order called covariance

$$\text{cov}(X, Y) = E((X - E(X))(Y - E(Y))) = E(XY) - E(X)E(Y).$$

In the coordinate form the covariance can be written down in the following form

$$\text{cov}(X, Y) = \sum_{i=1}^n \sum_{j=1}^m (x_i - \mu_X)(y_j - \mu_Y) p_{i,j},$$

where  $p_{i,j} = p(x_i, y_j)$ .

If for two random quantities of  $X$  and  $Y$  at increase of one random variable there is the tendency to increase of the other, then  $\text{cov}(X, Y) > 0$ ; If at increase of one random variable there is the tendency to decrease of the other, then  $\text{cov}(X, Y) < 0$ .

If the behavior is not predictable, then  $\text{cov}(X, Y) = 0$ . In this case we can say that random quantities are not correlated, but it does not mean that they are independent, even though, for independent random quantities  $\text{cov}(X, Y) = 0$ . The normalized covariance is called correlation (see eq. (A.3))

$$-1 \leq \text{cor}(X, Y) = \frac{\text{cov}(X, Y)}{\sqrt{\text{var}(X)\text{var}(Y)}} \leq 1 \quad (\text{A.3})$$

Let  $X = (X_1, X_2, \dots, X_n)$  be a vector. Its coordinates are random quantities, then

$$\begin{aligned} \text{cov}(X) &= \text{cov}(X_1, X_2, \dots, X_n) = \Sigma = E\left((X - u)(X - u)^T\right) = \\ &= \begin{pmatrix} E((X_1 - \mu_1)(X_1 - \mu_1)) & \cdots & E((X_n - \mu_n)(X_1 - \mu_1)) \\ \vdots & \ddots & \vdots \\ E((X_1 - \mu_1)(X_n - \mu_n)) & \cdots & E((X_n - \mu_n)(X_n - \mu_n)) \end{pmatrix} = \Sigma. \end{aligned}$$

The matrix  $\Sigma$  is called covariation matrix.

For the case of many variables the continuous  $n$ -dimensional random variable of  $X$  is distributed under the normal law  $N(\mu, \Sigma)$ , if its density function has the appearance

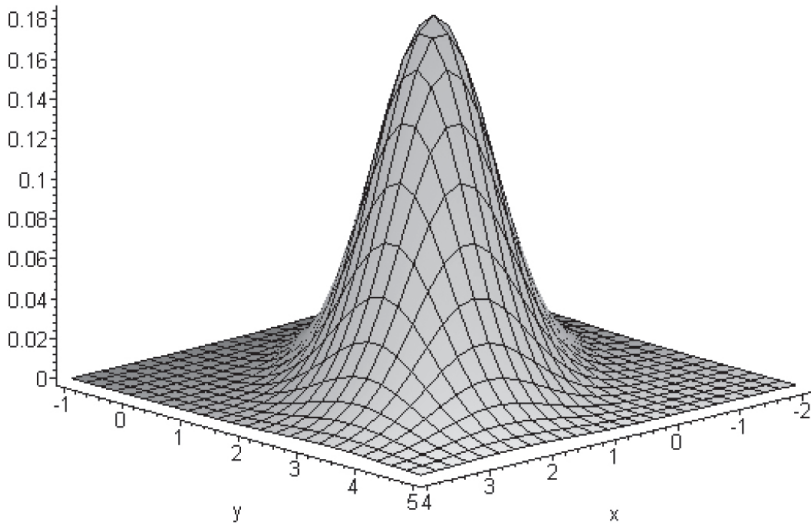
$$\begin{aligned} f(x) &= \frac{1}{(2\pi)^{n/2} |\Sigma|^{1/2}} \exp\left(-\frac{1}{2} \left((x - \mu)^T \Sigma^{-1} (x - \mu)\right)\right) = \\ &= \frac{1}{(2\pi)^{n/2} |\Sigma|^{1/2}} \exp\left(-\frac{1}{2} \left( (x_1 - \mu_1, \dots, x_n - \mu_n) \Sigma^{-1} \begin{pmatrix} x_1 - \mu_1 \\ \vdots \\ x_n - \mu_n \end{pmatrix} \right)\right). \end{aligned}$$

There is the covariation matrix  $\Sigma$ , its determinant  $|\Sigma|$  and its inverse  $\Sigma^{-1}$ .

The example of two-dimensional density function is presented in Figure A.3.

If all sizes  $(X_1, X_2, \dots, X_n)$  are independent, density function will have the following form

$$f(x) = \prod_{i=1}^n \frac{1}{\sigma_i \sqrt{2\pi}} \exp\left(-\frac{(x_i - \mu_i)^2}{2\sigma_i^2}\right).$$



**Fig. A.3.** The graph of density function  $N\left(\begin{pmatrix} 1 \\ 2 \end{pmatrix}, \begin{pmatrix} 1 & 0.5 \\ 0.5 & 1 \end{pmatrix}\right)$

Let  $\Phi$  be the matrix, which columns are combined eigenvectors of the matrix  $\Sigma$ , then (owing to the orthonormality)

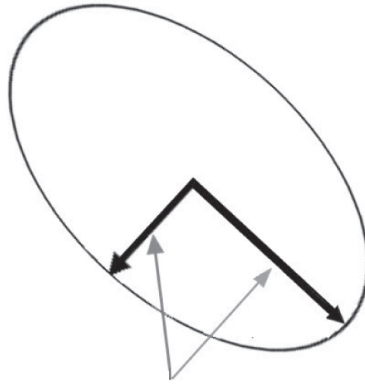
$$\Phi^{-1} = \Phi^T.$$

If  $\Sigma\Phi = \Phi\Lambda$ , where  $\Lambda$  is the scalar matrix with the corresponding eigenvalues of the matrix  $\Sigma$  on diagonal, then  $\Sigma = \Phi\Lambda\Phi^{-1}$  and, therefore  $\Sigma^{-1} = \Phi\Lambda^{-1}\Phi^{-1}$ . Through  $\Lambda^{-1/2}$  let's designate the matrix, such that  $\Lambda^{-1/2}\Lambda^{-1/2} = \Lambda^{-1}$ , then  $\Sigma^{-1} = (\Phi\Lambda^{-1/2})(\Phi\Lambda^{-1/2})^T = \Xi\Xi^T$ .

Thus

$$\begin{aligned} \left( (x-\mu)^T \Sigma^{-1} (x-\mu) \right) &= \left( (x-\mu)^T \Xi\Xi^T (x-\mu) \right) = \\ &= \left( \Xi^T (x-\mu) \right)^T \left( \Xi^T (x-\mu) \right) = \left| \Xi^T (x-\mu) \right|^2. \end{aligned}$$

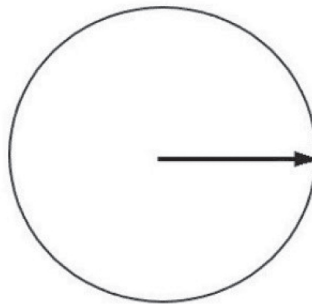
Noticing that the matrix  $\Xi$  represents the matrix of transformations (rotations and scaling), we receive that points  $x$ , meeting the condition  $\left| \Xi^T (x-\mu) \right|^2 \equiv \text{const}$  lie on the ellipse (see Fig. A.4).



Eigenvectors of the matrix  $\Sigma$

**Fig. A.4.** The set of points equidistant from the center in sense of Mahalanobis distance

Number  $\sqrt{\left((x-\mu)^T \Sigma^{-1} (x-\mu)\right)}$  is called Mahalanobis's distance between  $x$  and  $\mu$  (see Fig. A.4). In particular, if all sizes  $(X_1, X_2, \dots, X_n)$  are independent, the Mahalanobis's distance degenerates to Euclidean distance  $\sqrt{\left((x-\mu)^T (x-\mu)\right)}$  (see Fig. A.5).

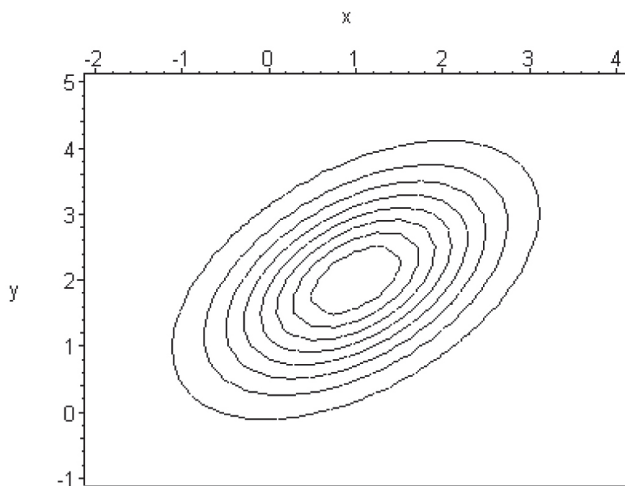


**Fig. A.5.** The set of points equidistant from the center in sense of Euclidean distance

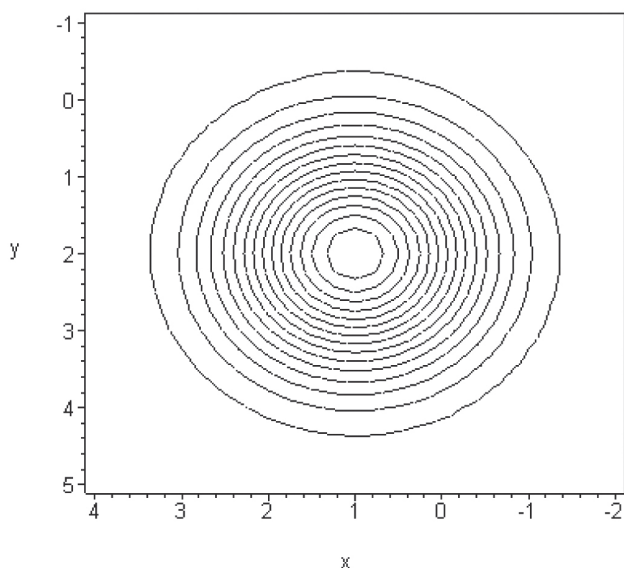
Let's give the example of two-dimensional Gaussian function.

Let  $\mu = (1, 2)$  and  $\Sigma = \begin{pmatrix} 1 & 0.5 \\ 0.5 & 1 \end{pmatrix}$ , then level lines of the corresponding function of Gauss will have the form presented in Figure A.6.

If  $\mu = (1, 2)$ ,  $\Sigma = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ , the representation is degenerate to spherical function of Gauss (see Fig. A.7).



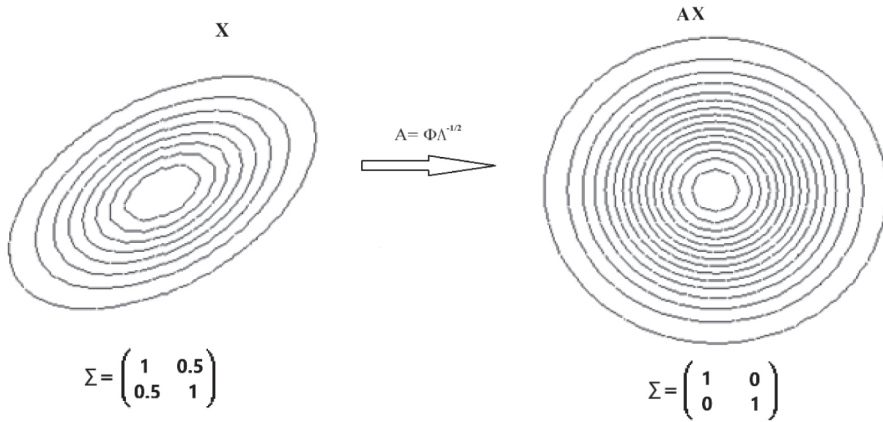
**Fig. A.6.** Isolines of Gaussian function with  $\mu = (1, 2)$ ,  $\Sigma = \begin{pmatrix} 1 & 0.5 \\ 0.5 & 1 \end{pmatrix}$



**Fig. A.7.** Isolines of the spherical function of Gauss at  $\mu = (1, 2)$ ,  $\Sigma = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$

If  $n$ -dimensional random variable  $X$  has density  $N(\mu, \Sigma)$ , size  $AX$  has density  $N(A^T\mu, A^T\Sigma A)$ , thus for any random variable of  $X$  it is possible to pick up the transformation transferring to the random variable with the spherical density function (see Fig. A.8).





**Fig. A.8.** Transformation of the random quantity with the density function with  $\Sigma = \begin{pmatrix} 1 & 0.5 \\ 0.5 & 1 \end{pmatrix}$  in the random quantity with the spherical function of density

More information relating to statistical analysis the reader can find for example in Berry, Browne 2006, Hand et al. 2001, Larose 2005, Press et al. 2007, Rao, Toutenburg 1999, Theodordis, Koutroumbas 2006, and Vapnik 2000.

## References

- Abdullah L. 2013. *Fuzzy multi criteria decision making and its applications: A brief review of category*, Procedia – Social and Behavioral Science, vol. 97, pp. 131–136, DOI: 10.1016/j.sbspro.2013.10.213.
- Aggarwal C.C., Reddy C.K. 2013. *Data Clustering: Algorithms and Applications, Series: Data Mining and Knowledge Discovery*, Chapman and Hall/CRC, August 21, 2013.
- Amorim R.C.D., Mirkin B. 2012. *Minkowski metric, feature weighting and anomalous cluster initializing in K-Means clustering*, Journal Pattern Recognition, vol. 45(3), March, 2012, pp. 1061–1075.
- Archambea C., Delannay N., Verleysen M. 2008. *Mixtures of robust probabilistic principal component analyzers*, Elsevier, Neurocomputing, vol. 71, no. 7–9, pp. 1274–1282.
- Avery O., MacLeod M., McCarthy M. 1944. *Studies on the Chemical Nature of the Substance Inducing Transformation of Pneumococcal Types: Induction of Transformation by a Deoxyribonucleic Acid Fraction Isolated from Pneumococcus Type III*, Journal of Experimental Medicine, vol. 79(2), pp. 137–158, DOI: 10.1084/jem.79.2.137, <http://jem.rupress.org/content/79/2/137/tab-pdf> (access: 9.04.2018).
- Babuska R., Van der Veen P.J., Kaymak U. 2002. *Improved Covarians Estimation for Gustafson Kessel Clustering*, Proceedings of the 2002 IEEE, Fuzzy Systems, pp. 1081–1084, <https://pure.tue.nl/ws/files/3635433/Metis256338.pdf> (access: 30.03.2018).
- Baker L.D., McCallum A.K. 1998. *Distributional Clustering of Words for Text Classification*, Proceedings of SIGIR-98, 21st ACM International Conference on Research and Development in Information Retrieval (Melbourne, Australia), pp. 96–103.
- Baldwin J.F., Xie D. 2005. *Simple Fuzzy Logic Rules Based on Fuzzy Decision Tree for Classification and Prediction Problem*, in: Shi Z., He Q. (ed.), *Intelligent Information Processing II. IIP 2004*, IFIP International Federation for Information Processing, vol. 163, Springer, Boston, MA, pp. 175–184, DOI: 10.1007/0-387-23152-8\_23.

- Banzhaf W., Francone F.D., Keller R.E., Nordin P. 1998. *Genetic programming: an introduction: on the automatic evolution of computer programs and its applications*, Morgan Kaufmann Publishers Inc. San Francisco, CA.
- Bara'a A. 2004. *Diploid GA with Real-valued vs. Binary-coded Recombination Operator in Diploid Genetic Algorithm*, Iraqi Journal of Science, pp. 255–260.
- Barbakh W.A., Wu Y., Fyfe C. 2009. *Review of Clustering Algorithms*, in: *Non-Standard Parameter Adaptation for Exploratory Data Analysis, Studies in Computational Intelligence*, vol. 249, Springer, Berlin, Heidelberg, pp. 7–28, DOI: 10.1007/978-3-642-04005-4\_2/ (access: 20.03.2018).
- Basu C., Hirsh H., Cohen W. 1998. *Recommendation as Classification: Using Social and Content-Based Information in Recommendation*, Proceedings of the Fifteenth National Conference on Artificial Intelligence, pp. 11–15.
- Bentley P.J., Wakefield J.P. 1996. *Overview of a Generic Evolutionary Design System*, in: *Proceedings of the 2nd On-line Workshop on Evolutionary Computation (WEC2)*, Nagoya University, Japan, pp. 53–56.
- Berry M., Browne M. 2006. *Lecture Notes in Data Mining*, Word Scientific.
- Bezdek J.C. 1981. *Pattern Recognition With Fuzzy Objective Function Algorithms*, New York, Plenum Press.
- Biesbroek R. 1999. *Genetic Algorithm Tutorial. 4.1 Mathematical foundations*.
- Björck A. 1996. *Numerical Methods for Least Squares Problems*, Series Other Titles in Applied Mathematics, Philadelphia.
- Bober M., Kucharski K., Skarbek W. 2003. *Face Recognition by Fisher and Scatter Linear Discriminant Analysis*, in: Petkov N., Westenberg M.A. (ed.), *Computer Analysis of Images and Patterns. CAIP 2003. Lecture Notes in Computer Science*, vol. 2756, Springer, Berlin–Heidelberg, pp. 638–645.
- Bock H.H. 1999. *Clustering and neural network approaches*, in: Gaul W., Locarek-Junge H. (ed.), *Classification in the Information Age*, Springer, Berlin–Heidelberg, pp. 42–57.
- Bonabeau E., Dorigo M., Theraulaz G. 1999. *Swarm Intelligence: From Natural to Artificial Intelligence*, Oxford University Press, New York.
- Bordes A., Ertekin S., Weston J., Bottou L. 2005. *Fast kernel classifiers with online and active learning*, Journal of Machine Learning Research, vol. 6, pp. 1579–1619.
- Borisov A.N., Krumberg O.A., Fyodorov I.P. 1990. *Decision Making Based on Fuzzy Models: Application Examples (Examples of models)*, Riga: Zinatne Press.
- Borowik B., Karpinsky M., Lahno V., Petrov O. 2013. *Theory of digital automata*, Springer Science+Business Media, International Series on Intelligent Systems, Control and Automation: Science and Engineering, vol. 63, DOI: 10.1007/978-94-007-5228-3.

- Boser B.E., Guyon I.M., Vapnik V.N. 1992. *A Training Algorithm for Optimal Margin Classifiers*, in: D. Haussler (ed.), *Proceedings of the Fifth Annual ACM Workshop on Computational learning theory COLT*, Pittsburgh, Pennsylvania, USA, Proceedings of the fifth annual workshop on Computational learning theory – COLT '92, pp. 144–152, DOI: 10.1145/130385.130401.
- Brusilovsky P., Kobsa A., Nejdl W. (ed.). 2007. *The Adaptive Web. Methods and Strategies of Web Personalization*, Springer, Berlin – Heidelberg.
- Building Classification Models: ID3 and C4.5*, <https://cis.temple.edu/~giorgio/cis587/readings/id3-c45.html> (access: 20.03.2018).
- Burges C.J. 1998. *A Tutorial on Support Vector Machines for Pattern Recognition*, *Data Mining and Knowledge Discovery*, vol. 2(2), pp. 121–167, DOI: 10.1023/A:1009715923555 (access: 20.03.2018).
- Chakrabarti S. 2004. *Mining The Web Discovering Knowledge From Hypertext Data*, Morgan Kaufmann Publishers.
- Chan P.K., Schlag M.D.F., Zien J.Y. 1994. *Spectral K-way Ratio-cut Partitioning and Clustering*, *IEEE Trans. Computer-Aided Design*, vol. 13, pp. 1088–1096.
- Chapelle O., Vapnik V. 1999. *Model selection for support vector machines*, in: *Advances in Neural Information Processing Systems*, vol. 12, Cambridge, Mass: MIT Press, <http://papers.nips.cc/paper/1663-model-selection-for-support-vector-machines.pdf> pp. 230–236 (access: 18.04.2018).
- Charu C.A. 2015. *Data Mining, The Textbook*, Springer International Publishing, DOI: 10.1007/978-3-319-14142-8.
- Chernov V., Dorokhov O., Dorokhova L., Chubuk V. 2015. *Using Fuzzy Logic for Solution of Economic tasks: Two examples of Decision Making Under Uncertainty*, *Montenegrin Journal of Economics*, vol. 11, no. 1, pp. 85–100.
- Chernov V., Dorokhova L., Dorokhov O. 2016. *Fuzzy Approach to Estimates Entropy and Risks for Innovative Projects and Programs*, *Montenegrin Journal of Economics*, vol. 12, no. 3, pp. 55–68, [http://www.mnje.com/sites/mnje.com/files/multimedia/main\\_pages/files/2012/03/55-68\\_dorokhov.pdf](http://www.mnje.com/sites/mnje.com/files/multimedia/main_pages/files/2012/03/55-68_dorokhov.pdf) (access: 20.03.2018).
- Chiu C.Y., Park S.C. 1994. *Fuzzy cash flow analysis using present worth criterion*, *The Engineering Economist*, vol. 39(2), pp. 113–138.
- Cohen W.W., Fan W. 2000. *Web-collaborative filtering: recommending music by crawling the Web*, *Computer Networks*, vol. 33, no. 1–6, pp. 685–698.
- Collaborative filtering, [http://en.wikipedia.org/wiki/Collaborative\\_filtering](http://en.wikipedia.org/wiki/Collaborative_filtering) (access: 20.03.2018).
- Cortes C., Vapnik V. 1995. *Support-vector networks*, *Machine Learning*, vol. 20(3), pp. 273–297, DOI: 10.1007/BF00994018.

- Cramer N.L. 1985. *A Representation for the Adaptive Generation of Simple Sequential Programs*, Proceedings of the 1st International Conference on Genetic Algorithms, Pittsburgh, 24–26 July 1985, pp.183–187.
- Cristianini N., Shawe-Taylor J. 2000. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*, Cambridge University Press, New York.
- De Jong K.A. 1975. *An analysis of the behavior of a class of genetic adaptive systems*, Unpublished PhD thesis. – University of Michigan, Ann Arbor, University Microfilms No. 76-9381.
- De Jong K.A., Spears W.M. 1991. *An analysis of the interacting roles of population size and crossover in genetic algorithms*, in: Schwefel H.P., Männer R. (ed.), *Parallel Problem Solving from Nature*. PPSN 1990. Lecture Notes in Computer Science, vol. 496, pp. 38–47, Springer, Berlin–Heidelberg.
- De Jong K.A., Spears W.M. 1992. *A formal analysis of the role of multi-point crossover in genetic algorithms*, Annals of Mathematics and Artificial Intelligence vol. 5: 1, pp. 1–26, DOI: 10.1007/BF01530777 (access: 20.03.2018).
- Deb K., Agrawal R.B. 1999. *Understanding interaction among genetic algorithms parameters*, Foundation of Genetic Algorithms, vol. 5, Banzhaf W., Reeves C., (ed.), Morgan Kaufmann.
- Delannay N., Verleysen M. 2008. *Collaborative filtering with interlaced generalized linear models*, Neurocomputing, Vol. 71, Issues 7–9, pp. 1300–1310.
- Dempster A.P., Laird N.M., Rubin D.B. 1977. *Maximum Likelihood from Incomplete Data via the EM Algorithm*, Journal of the Royal Statistical Society, Series B (Methodological), vol. 39, no. 1, pp. 1–38.
- Denebourg J.L., Pasteels J.M., Verhaeghe J.C. 1993. *Probabilistic Behavior in Ants: a Strategy of Errors?*, Journal of Theoretical Biology, vol. 105, pp. 259–271.
- Dhillon I., Guan Y., Kulis B. 2004. *Kernel k-means, spectral clustering and normalized cuts*, Proceedings of the tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Seattle, WA, USA.
- Diana Algorithm* [http://www.dpye.iimas.unam.mx/lety/archivos/multivariado2018\\_1/clases/cluster/Divisive%20Analysis%20\(Diana\).pdf](http://www.dpye.iimas.unam.mx/lety/archivos/multivariado2018_1/clases/cluster/Divisive%20Analysis%20(Diana).pdf) (access: 20.03.2018).
- Diaz-Gomez P.A., Hougen D.F. 2007. *Initial population for genetics algorithms: A metric approach*, Proceedings of the International Conference on Genetic and Evolutionary Methods, pp. 43–49.
- Diday E., Simon J.C. 1976. *Clustering Analysis*, in: Fu K.S. (ed.), *Digital Pattern Recognition. Communication and Cybernetics*, vol. 10, Springer, Berlin–Heidelberg, pp. 47–94.
- Domingos P., Pazzani M. 1997. *On the Optimality of the Simple Bayesian Classifier under Zero-One Loss*, Machine Learning, vol. 29, no. 2–3, pp. 103–130, DOI: 10.1023/A:1007413511361.

- Dorigo M. 1992. *Optimization, learning and natural algorithms*, PhD Thesis, Politecnico di Milano.
- Dorigo M., Stützle T. 2004. *Ant colony optimization*, MIT Press, Cambridge.
- Dua D., Karra Taniskidou E. 2017. *UCI Machine Learning Repository* [<http://archive.ics.uci.edu/ml>], Irvine, CA: University of California, School of Information and Computer Science, <http://archive.ics.uci.edu/ml> (access: 20.07.2017).
- Dubois D., Prade H. 1988. *Possibility Theory: An Approach to Computerized Processing of Uncertainty*, New York, Plenum Press.
- Dubois M. 1993. *Readings in Fuzzy Sets for Intelligent Systems*, Dubois D., Prade M. (ed.), Morgan Kaufmann: San Francisco, CA, USA.
- Duda J., Szydło S. 2011. *Collective intelligence of genetic programming for macroeconomic forecasting*, Proceedings of the Third international conference on Computational collective intelligence: technologies and applications, pp. 445–454, 21–23 September 2011, Gdynia, Poland.
- Dvorský J. 2004. *Word-based Compression Methods for Information Retrieval Systems*, Ph.D. thesis, Charles University, Prague.
- Efstathiou J., Rajkovic V. 1980. *Multi-attribute decision-making using a fuzzy, heuristic approach*, IEEE International Journal of Man-Machine Studies, vol. 12(2), pp. 141–156, DOI: 10.1016/S0020-7373(80)80014-9 (access: 20.03.2018).
- Ekel P.Ya. 2002. *Fuzzy sets and models of decision making*, Computers & Mathematics with Applications, vol. 44(7), October 2002, pp. 863–875, DOI: 10.1016/S0898-1221(02)00199-2 (access: 20.03.2018).
- Encyclopedia Britannica. 2018. *Least squares method*, R. Routledge, <https://www.britannica.com/topic/least-squares-approximation> (access: 20.01.2018).
- Evans L., Lohse N. 2011. *Optimized Fuzzy Decision Tree Data Mining for Engineering Applications*, in: Perner P. (ed.), *Advances in Data Mining. Applications and Theoretical Aspects*, ICDM 2011, Lecture Notes in Computer Science, vol. 6870, Springer, Berlin–Heidelberg, pp. 228–239.
- Everitt B.S., Landau S., Leese M. 2001. *Cluster Analysis (4th edition)*, Arnold, London.
- Fedrizzi M., Pasi G. 2008. *Fuzzy Logic Approaches to Consensus Modelling in Group Decision Making*, Studies in Computational Intelligence (SCI), vol. 117, Springer, Berlin–Heidelberg, pp. 19–37.
- Ferreira C. 2006. *Automatically Defined Functions in Gene Expression Programming*, in: Nedjah N., de Mourelle L.M., Abraham A. (ed.), *Genetic Systems Programming: Theory and Experiences*, Studies in Computational Intelligence, vol. 13, pp. 21–56, Springer, Berlin.
- Fisher R.A. 1936. *The use of multiple measurements in taxonomic problems*, Annals of Eugenics, vol. 7, pp. 111–132.

- Fogel L.J. 1999. *Intelligence through simulated evolution: forty years of evolutionary programming*, John Wiley & Sons, Inc. New York.
- Fogel L.J., Owens A.J., Walsh M.J. 1966. *Artificial Intelligence through Simulated Evolution*, John Wiley & Sons, Inc. New York.
- Fraley C., Raftery A.E. 1998. *How many clusters? Which clustering method? Answers via model-based cluster analysis*, *The Computer Journal*, vol. 41, no. 8, pp. 578–588.
- Fredericks E.M., Cheng B.H.C. 2013. *Exploring automated software composition with genetic programming*, *Proceeding of the Fifteenth Annual Conference Companion on Genetic and Evolutionary Computation Conference Companion*, July 06.10.2013, Amsterdam.
- Gambardella L.M., Dorigo M. 2000. *An ant colony system hybridized with a new local search for the sequential ordering problem*, *INFORMS Journal on Computing*, vol. 12(3), pp. 237–255.
- Ghazanfar M.A., Prügel-Bennett A. 2010. *An improved switching hybrid recommender system using Naive Bayes classifier and collaborative filtering*, in: *Lecture Notes in Engineering and Computer Science: Proceedings of The International Multi Conference of Engineers and Computer Scientists 2010*, IMECS 2010, 17–19 March, 2010, Hong Kong, pp. 493–502.
- Goldberg D.E. 1989. *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA.
- Goldberg D.E., Sastry K. 2001. *A practical schema theorem for genetic algorithm design and tuning*, *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 328–335.
- Gorban A.N., Kégl B., Wunsch D.C., Zinovyev A. (ed.). 2008. *Principal Manifolds for Data Visualization and Dimension Reduction*, *Series Lecture Notes in Computational Science and Engineering*, Springer, Berlin–Heidelberg, DOI: 10.1007/978-3-540-73750-6.
- Gorban A.N., Mirkes E.M., Zinovyev A. 2016. *Piece-wise Quadratic Approximations of Arbitrary Error Functions for Fast and Robust Machine Learning*, *Neural Networks*, vol. 84, pp. 28–38, DOI: <https://doi.org/10.1016/j.neunet.2016.08.007>.
- Gordon A.D. 1999. *Classification, Monographs on Statistics & Applied Probability*, CRC Press.
- Gordon D.M. 1999. *Ants at Work. How an Insect Society is Organized*, Norton and Company Inc., New York.
- Goss S., Aron S., Deneubourg J.L., Pasteels J.M. 1989. *Self-organized Shortcuts in the Argentine Ant*, *Naturwissenschaften*, vol. 76, pp. 579–581, Springer.



- Grabmeier J., Rudolph A. 2002. *Techniques of Cluster Algorithms in Data Mining, Data Mining and Knowledge Discovery*, vol. 6(4), pp. 303–360, DOI: 10.1023/A:1016308404627.
- Hamel L. 2009. *Knowledge Discovery With Support Vector Machines*, Wiley Series on Methods and Applications in Data Mining, Hoboken, New Jersey, John Wiley and Sons Inc.
- Han J., Kamber M., Pei J. 2011. *Data Mining: Concepts and Techniques*, 3d edition, Morgan Kaufmann Publishers.
- Hand D.J., Mannila H., Smyth P. 2001. *Principles of Data Mining*, A Bradford Book.
- Hartigan J. 1975. *Clustering algorithms*, John Willey and Sons.
- Hastie T., Tibshirani R., Friedman J. 2009. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Springer Series in Statistics, Springer, DOI: 10.1007/978-0-387-84858-7 (access: 20.03.2018).
- Herlocker J.L., Konstan J.A., Terveen L.G., Riedl J.T. 2004. *Evaluating collaborative filtering recommender systems*, ACM Transactions on Information Systems (TOIS) archive, Vol. 22, pp. 734–749, <https://groupLens.org/site-content/uploads/evaluating-TOIS-20041.pdf> (access: 20.05.2018).
- Hill W.C., Stead L., Rosenstain M., Furnas G.W. 1995. *Recommending and Evaluating Choices in a Virtual Community of Use*, Conference Paper.
- Hoffmann T., Scholkopf B., Smola A.J. 2008. Kernel Methods in Machine Learning, *The Annals of Statistics*, vol. 36, no. 3, pp. 1171–1220, Institute of Mathematical Statistics, DOI: 10.1214/009053607000000677.
- Hofmann T. 2004. *Latent semantic models for collaborative filtering*, ACM Transactions on Information Systems (TOIS), vol. 22, no. 1, pp. 89–115, January 2004, DOI: 10.1145/963770.963774.
- Holland J.H. 1994. *Adaptation in natural and artificial systems, An introductory analysis with application to biology, control, and artificial intelligence*, London, MIT Press, A Bradford Book.
- Iatan I.F. 2010. *The Fisher's Linear Discriminant*, in: Borgelt C. et al. (ed.), *Combining Soft Computing and Statistical Methods in Data Analysis*, Series Advances in Intelligent and Soft Computing, vol. 77, Springer, Berlin–Heidelberg, pp. 345–352.
- Idris I. 2014. *Python Data Analysis*, Packt Publishing Ltd., October 2014.
- Intan R., Yuliana O.Y. 2009. *Fuzzy Decision Tree Induction Approach for Mining Fuzzy Association Rules*, in: Leung C.S., Lee M., Chan J.H. (ed.), *Neural Information Processing. ICONIP 2009. Lecture Notes in Computer Science*, vol. 5864, Springer, Berlin–Heidelberg, pp. 720–728.
- Jain A. 2010. *Data clustering: 50 years beyond k-means*, *Pattern Recognition Letters*, vol. 31, pp. 651–666.



- Jain A.K., Murty M.N., Flynn P.J. 1999. *Data clustering: a review*, ACM Computing Surveys, vol. 31(3), pp. 264–323.
- Janikow C.Z., Michalewicz Z., *An experimental Comparison of Binary and Floating Point Representation in Genetic Algorithms*, [http://umsl.edu/divisions/artscience/math\\_cs/about/People/Faculty/CezaryJanikow/folder%20two/Experimental.pdf](http://umsl.edu/divisions/artscience/math_cs/about/People/Faculty/CezaryJanikow/folder%20two/Experimental.pdf) (access: 20.03.2018).
- Jenssen R., Eltoft T., Principe J.C. 2004. *Information theoretic spectral clustering*, *Proceedings of Neural Networks 2004*. Proceedings. 2004 IEEE International Joint Conference on Neural Networks, vol. 1, pp. 111–116.
- Joachims T. 2009. *Retrospective on Transductive Inference for Text Classification using Support Vector Machines*, Proceedings of the International Conference on Machine Learning (ICML), 1999–2009.
- Joachims T., *SVM-light Support Vector Machine*, <http://svmlight.joachims.org/> (access: 20.03.2018).
- John G.H., Langley P. 1995. *Estimating continuous distributions in Bayesian classifiers*, in: Besnard P., Hanks S. (ed.), *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, San Francisco, CA: Morgan Kaufmann, pp. 338–345.
- Jolliffe I.T. 2002. *Principal Component Analysis*, *Springer Series in Statistics*, Springer, New York, DOI: 10.1007/b98835.
- Kannan R., Vempala S., Veta A. 2000. *On Clusterings – Good, Bad and Spectral*, *Proceedings 41st Annual Symposium on Foundations of Computer Science*, Redondo Beach, CA, USA, IEEE, pp. 367–377, DOI: 10.1109/SFCS.2000.892125.
- Kaufman L., Rousseeuw P.J. 1990. *Finding groups in data. An Introduction to Cluster Analysis*, Wiley Series in Probability and Statistics, John Wiley & Sons, New York.
- Kaufmann A. 1975. *Introduction to the Theory of Fuzzy Subsets*, vol. 1, Fundamental Theoretical Elements, Academic Press, New York.
- Kennedy J. 2010. *Particle Swarm Optimization*, in: Sammut C., Webb G.I. (ed.), *Encyclopedia of Machine Learning*, Springer, Boston, MA, pp. 760–766.
- Kohavi J., Backer B., Sommerfield D. 1997. *Improving simple Bayes*, *Technical report*, Data Mining and Visualization Group, Silicon Graphics.
- Kohavi R. 1996. *Scaling up the accuracy of naive-Bayes classifiers: A decision-tree hybrid*, Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, vol. 7, pp. 202–207.
- Kohonen T. 1982. *Clustering, taxonomy, and topological maps of patterns*, Proceedings of the sixth international conference on pattern recognition, Silver Spring MD: IEEE Computer Society Press, pp. 114–128.

- Kohonen T. 1990. *The self-organizing map*, Proceedings of the IEEE, vol. 78(9), pp. 1464–1480.
- Kohonen T. 2001. *Self-organizing maps*, Springer Series in Information Sciences, vol. 30, Springer, Berlin–Heidelberg Inc., DOI: 10.1007/978-3-642-56927-2.
- Kohonen T. 2013. *Essentials of the self-organizing map*, Neural Networks, vol. 37, pp. 52–65, <https://pdfs.semanticscholar.org/4e9b/467deb651f1333f4a99331c50760286f452e.pdf> (access: 26.04.2018).
- Kohonen, T., Somervuo P. 1998. *Self-organizing maps of symbol strings*, Neurocomputing, vol. 21(1–3), 6 November 1998, pp. 19–30, DOI: [https://doi.org/10.1016/S0925-2312\(98\)00031-9](https://doi.org/10.1016/S0925-2312(98)00031-9).
- Koren Y., Bell R.M., Volinsky C. 2009. *Matrix factorization techniques for recommender systems*, IEEE Computer, vol. 42, no. 8, pp. 30–37.
- Koza J.R. 1989. *Hierarchical genetic algorithms operating on populations of computer programs*, Proceedings of the 11th International Joint Conference on Artificial Intelligence, August 20–25, 1989, Detroit, Michigan, pp. 768–774.
- Koza J.R. 1992. *Genetic programming: on the programming of computers by means of natural selection*, MIT Press, Cambridge, MA.
- Koza J.R. 1994. *Genetic programming for economic modeling*, Statistics and Computing, vol. 4(2), pp. 187–197.
- Kruskal J.B., Wish M. 1978. *Multidimensional Scaling*, Sage University Paper Series on Quantitative Applications in the Social Sciences, No. 07-011, Sage Publications, Newbury Park, DOI: 10.4135/9781412985130.
- Kuppussamy K.S., Aghila G. 2011. *A Personalized Web Page Content Filtering Model Based on Segmentation*, International Journal of Information Sciences and Techniques (IJIST), Vol. 2, No. 1 January 2011, pp. 41–51.
- Lance G.N., Williams W.T. 1967. *A general theory of classificatory sorting strategies: I. Hierarchical Systems*, The Computer Journal, vol. 9, pp. 373–380.
- Lance G.N., Williams W.T. 1968. *Mixed-data classificatory programs I. Agglomerative systems*, The Australian Computer Journal, vol. 1, pp. 15–20.
- Lance G.N., Williams W.T. 1968. *Mixed-data classificatory programs I. Divisive systems*, The Australian Computer Journal, vol. 1, pp. 82–85.
- Langley P., Iba W., Thompson K., *An analysis of Bayesian classifiers*, Proceedings of the tenth national conference on Artificial intelligence, July 12–16, 1992, San Jose, California, pp. 223–228.
- Langone R., Mall R., Alzate C., Suykens J.A.K. 2016. *Kernel Spectral clustering and applications*, in: Celebi M.E., Aydin K. (ed.), *Unsupervised Learning Algorithms*, Springer International Publishing, DOI: 10.1007/978-3-319-24211-8.

- Larose D.T. 2005. *Discovering Knowledge in Data: An Introduction to Data Mining*, Wiley, New York.
- Larsen P.M. 1980. *Industrial applications of fuzzy logic control*, International Journal of Man-Machine Studies, vol. 12(1), pp. 3–10, DOI: 10.1016/S0020-7373(80)80050-2.
- Lawson C.L. 1987. *Solving Least Squares Problems*, Series Classics in Applied Mathematics, vol. 15, Society for Industrial Mathematics, Philadelphia.
- Layton R. 2017. *Learning Data Mining with Python*, Packt Publishing Ltd., Birmingham – Mumbai.
- Lee C. 1990. *Fuzzy logic in control systems: Fuzzy logic controller – Part II*, IEEE Transactions on Systems, Man, and Cybernetics, vol. 20, no. 2, pp. 419–435.
- Li Y.-Z., Wang Y.-H. 2005. *PCA and algorithms analysis*, Journal of Suzhou University, vol. 1, pp. 32–36.
- Lim Y.J., Teh Y.W. 2007. *Variational Bayesian approach to movie rating prediction*, Proceedings of KDD Cup and Workshop, 2007, pp. 15–21, <https://www.cs.uic.edu/~liub/KDD-cup-2007/proceedings/variational-Lim.pdf> (access: 08.05.2018).
- Macnaughton-Smith P., Williams W.T., Dale M.B., Mockett L.G. 1964. *Dissimilarity analysis: a new technique of hierarchical sub-division*, Nature, vol. 202, pp. 1034–1035.
- Mamdani E.H. 1977. *Application of fuzzy logic to approximate reasoning using linguistic synthesis*, IEEE Transactions on Computers, vol. C-26, no. 12, pp. 1182–1191.
- Mamdani E.H., Assilian S. 1975. *An experiment in linguistic synthesis with a fuzzy logic controller*, International Journal of Man-Machine Studies, vol. 7(1), pp. 1–13, DOI: 10.1016/S0020-7373(75)80002-2 (access: 20.03.2018).
- Manukyan N., Eppstein M.J., Rizzo D.M. 2011. *Improved Cluster identification and Visualization in high-dimensional Data using Self-Organizing Maps*, Conference Paper and Presentation, EOS Transactions, American Geophysical Union Fall Meeting, San Francisco, CA, <http://adsabs.harvard.edu/abs/2011AGUFM.H34D..07M> (access: 26.04.2018).
- Marlin B. 2004. *Modeling user rating profiles for collaborative filtering*, in: Thrun S., Saul L., Schölkopf B. (ed.), *Advances in Neural Information Processing Systems*, vol. 16, Cambridge, MA, 2004. MIT Press, <https://pdfs.semanticscholar.org/efd9/ea3a7630bfe793dcf3747d98afc02b2b14dc.pdf> (access: 20.03.2018).
- Miller S.J. 2006. *The Method of Least Squares*, in Brown University, [https://web.williams.edu/Mathematics/sjmiller/public\\_html/BrownClasses/54/handouts/MethodLeastSquares.pdf](https://web.williams.edu/Mathematics/sjmiller/public_html/BrownClasses/54/handouts/MethodLeastSquares.pdf) (access: 20.01.2018).
- Milligan G.W., Cooper M.C. 1985. *An examination of procedures for determining the number of clusters in a data set*, Psychometrika, vol. 50, pp. 159–179.
- Milligan G.W., Cooper M.C. 1988. *A study of standardization of the variables in cluster analysis*, Journal of Classification, vol. 5, pp. 181–204.

- Mills R.T., Kumar J., Hoffman F.M., Hargrove W.W., Spruce J.P., Norman S.P. 2013. *Identification and Visualization of Dominant Patterns and Anomalies in Remotely Sensed Vegetation Phenology Using a Parallel Tool for Principal Components Analysis*, *Procedia Computer Science, ICCS*, vol. 18, pp. 2396–2405, [https://www.srs.fs.usda.gov/pubs/ja/2013/ja\\_2013\\_mills\\_001.pdf](https://www.srs.fs.usda.gov/pubs/ja/2013/ja_2013_mills_001.pdf) (access: 26.04.2018).
- Mirkin B. 2005. *Clustering for Data Mining. A Data Recovery Approach*, Chapman & Hall/CRC, London.
- Mitchell M. 1996. *Introduction to Genetic Algorithms*, MIT Press, Cambridge, MA.
- Naive Bayes algorithm for learning to classify text, <http://www.cs.cmu.edu/afs/cs/project/theo-11/www/naive-bayes.html> (access: 20.03.2018).
- Negnevitsky M. 2005. *Artificial Intelligence: A Guide to Intelligent Systems*, Pearson Education.
- Ng A.Y., Jordan M.I., Weiss Y. 2002. *On Spectral Clustering: Analysis and an algorithm*, in: Dietterich T., Becker S., Ghahramani Z. (ed.), *Neural Information Processing Systems, no. 14*, <https://papers.nips.cc/paper/2092-on-spectral-clustering-analysis-and-an-algorithm.pdf>, pp. 849–856 (access: 23.04.2018).
- Nguyen D.T., Doan H. 2012. *An Approach to Determine the Number of Clusters for Clustering Algorithms*, in: Nguyen N.T., Hoang K., Jędrzejowicz P. (ed.), *Computational Collective Intelligence, Technologies and Applications, ICCCI 2012. Lecture Notes in Computer Science*, vol. 765, pp. 485–494, Springer, Berlin, Heidelberg, DOI: 10.1007/978-3-642-34630-9\_50.
- Olaru C., Wehenkel L. 2003. *A complete fuzzy decision tree technique*, *Fuzzy Sets and Systems*, vol. 138(2), 1, pp. 221–254, DOI: 10.1016/S0165-0114(03)00089-7 (access: 20.03.2018).
- Olshen R.A., Rajaratnam B. 2010. *Successive Normalization of Rectangular Arrays*, *The Annals of Statistics*, vol. 38, no. 3, pp. 1638–1664, DOI: 10.1214/09-AOS743.
- Ordinary least squares method, Wikipedia, [https://en.wikipedia.org/wiki/Ordinary\\_least\\_squares](https://en.wikipedia.org/wiki/Ordinary_least_squares) (access: 20.01.2018).
- Pastukhov A.A., Prokofiev A.A. 2016. *Kohonen self-organizing map application to representative sample formation in the training of the multilayer perceptron*, *St. Petersburg Polytechnical University Journal: Physics and Mathematics*, vol. 2(2), June 2016, pp. 134–143, DOI: <https://doi.org/10.1016/j.spjpm.2016.05.012>, <https://www.sciencedirect.com/science/article/pii/S2405722316300809#cebib1> (access: 26.04.2018).
- Pazzani M. 1996. *Constructive induction of cartesian product attributes*, *Information, Statistics and Induction in Science*, pp. 66–77, World Scientific, <https://pdfs.semanticscholar.org/3208/ed3d4ff2de382ad6a16431cfe7118c000725.pdf>; (access: 20.03.2018).

- Pedregosa F., Varoquaux G., Gramfort A., Michel V., Thirion B., Grisel O., Blondel M., Prettenhofer P., Weiss R., Dubourg V., Vanderplas J., Passos A., Cournapeau D., Brucher M., Perrot M., Duchesnay É. 2011. *Scikit-learn: Machine Learning in Python*, JMLR, vol. 12, pp. 2825–2830.
- Pedrycz W. 1984. *An identification of fuzzy relational systems*, *Fuzzy Sets and Systems*, vol. 13, pp. 153–167.
- Penn B.S. 2005. *Using Self-organizing Maps to Visualize High-dimensional Data*, *Journal of Computers & Geosciences Archive*, vol. 31(5), June, 2005, pp. 531–544, Pergamon Press, Inc. Tarrytown, NY, USA, DOI: 10.1016/j.cageo.2004.10.009.
- Perkins J. 2014. *Python 3 Text Processing with NLTK 3 Cookbook*, Packt Publishing Ltd.
- Pestunov I.A., Berikov V.B., Kulikova E.A., Rylov S.A. 2011. *Ensemble of Clustering Algorithms for Large Datasets*, *Optoelectronics, Instrumentation and Data Processing*, vol. 47, no. 3, Allerton Press, pp. 245–252, url: <http://math.nsc.ru/AP/datamine/eng/oidp245.pdf> (access: 20.03.2018).
- Petrov A., Khoroshko V., Scherbak L., Petrov A., Aleksander M. 2016. *Reliability basics of information systems*, Petrov A. (ed.), AGH University of Science and Technology Press, Krakow.
- Poncelet P., Teisseire M., Masseglia F. (ed.). 2008. *Data Mining Patterns: New Methods and Applications*, *Information science reference*, Hershey, New York.
- Press W., Teukolsky S., Vetterling W., Flannery B. 2007. *Numerical Recipes: The Art of Scientific Computing*, Cambridge University Press, New York.
- Rana S. 1999. *Examining the Role of Local Optima and Schema Processing in Genetic Search*, PhD thesis, Colorado State University, Colorado.
- Rao C.R., Toutenburg H. 1999. *Linear Models, Least Squares and Alternatives*, Springer Series in Statistics, Springer, New York, DOI: 10.1007/b98889.
- Raschka S. 2015. *Python Machine Learning*, Packt Publishing Ltd., September 2015.
- Reiten T.E. 2017. *Classification with Multiple Classes using Naïve Bayes and Text Generation with a Small Data Set using a Recurrent Neural Network*, Master thesis, University of Agder, Faculty of Science and Technology Department of ICT.
- Resnick P., Varian H.R. 1997. *Recommender systems – introduction to the special section*, *Communications of the ACM*, vol. 40(3), pp. 56–58, DOI: 10.1145/245108.245121.
- Reuters-21578 text categorization test collection*, <http://www.daviddlewis.com/resources/testcollections/reuters21578/> (access: 20.03.2018).
- Ricci F., Rokach L., Shapira B., Kantor P. (ed.). 2011. *Recommender Systems Handbook*, Springer, Boston, MA, DOI: 10.1007/978-0-387-85820-3\_1.
- Richert W., Coelho L.P. 2013. *Building machine learning system with python*, Packt Publishing Ltd.

- Robinson E. 1981. *Least Squares Regression Analysis in Terms of Linear Algebra*, Published 1981 by Goose Pond Press in Houston.
- Rocchio J. 1966. *Document Retrieval Systems: Optimization and Evaluation*, Ph.D. Thesis, Harvard University.
- Rocchio J. 1971. *Chapter 14: Relevance feedback in information retrieval*, in: Salton G. (ed.), *The SMART retrieval system – experiments in automatic document processing*, Englewood Cliffs, NJ: Prentice-Hall Inc., pp. 313–323.
- Roiger R.J., Geatz M.W. 2003. *Data Mining: A Tutorial-Based Primer*, Addison Wesley, Pearson Education, Inc.
- Rokach L. 2009. *A survey of Clustering Algorithms*, in: Maimon O., Rokach L. (ed.), *Data Mining and Knowledge Discovery Handbook*, Springer, Boston, DOI: 10.1007/978-0-387-09823-4\_14.
- Rokach L., Maimon O. 2014. *Data mining with decision trees: Theory and Applications*, Series Machine Perception and Artificial Intelligence, World Scientific Publishing, Hackensack, vol. 81.
- Rossi F., Villa N. 2006. *Support vector machine for functional data classification*, *Neurocomputing*, vol. 69(7–9), pp. 730–742, <https://doi.org/10.1016/j.neucom.2005.12.010> (access: 20.03.2018).
- Roux M. 2015. *A comparative study of divisive hierarchical clustering algorithms*, *Ithaca: Cornell University Library*, Computer Science, Data Structures and Algorithms, version 2, <https://arxiv.org/abs/1506.08977> (access 20.03.2018).
- Salton G., Buckley C. 1988. *Term weighting approaches in automatic text retrieval*, *Information Processing and Management*, vol. 24(5), pp. 513–523.
- Sammur C., Webb G.I. (ed.). 2017. *Encyclopedia of Machine Learning and data mining*, Springer, New York.
- Sarwar B., Karypis G., Konstan J., Riedl J. 2000. *Application of Dimensionality Reduction in Recommender Systems. A Case Study*, Proceedings ACM WebKDD Workshop, Boston, MA, USA, Technical Report.
- Sarwar B., Karypis G., Konstan J., Riedl J. 2001. *Item-Based Collaborative Filtering Recommendation Algorithms*, Proceedings of the 10th International WWW Conference, Hong Kong, China, pp. 285–295.
- Schafer J.B., Konstan, J.A., and Riedl, J. 1999. *Recommender Systems in E-Commerce*, ACM Conference on Electronic Commerce (EC-99), pp. 158–166.
- Scholarpedia. *Kohonen network*, [http://www.scholarpedia.org/article/Kohonen\\_network](http://www.scholarpedia.org/article/Kohonen_network) (access: 20.03.2018).
- Scholkopf B., Burges J.C., Smola A.J., Muller K.R. 1998. *Kernel Principal Component Analysis*, in: Scholkopf B., Burges J.C., Smola A.J., Muller K.R. (ed.), *Advances in Kernel Methods – Support Vector Learning*, MIT Press, Cambridge, MA, Chapter 20, pp. 327–352.



- Sebastiani F. 2002. *Machine learning in automated text categorization*, ACM Computing Surveys (CSUR), vol. 34, no. 1, pp. 1–47, DOI: 10.1145/505282.505283.
- Self-organizing map/Wikipedia, [https://en.wikipedia.org/wiki/Self-organizing\\_map](https://en.wikipedia.org/wiki/Self-organizing_map) (access: 20.03.2018).
- Shin C.Y., Wang P.P. 2010. *Economic application of fuzzy subset theory and fuzzy logic: A brief survey*, New Mathematics and Natural Computation, vol. 6, no. 3, pp. 301–320, DOI: 10.1142/S1793005710001773 (access: 20.03.2018).
- Shumeyko A.A., Sotnik S.L. 2009. *Using of Genetic Algorithms for Text Classification Problems*, Annals. Computer Science Series, vol. 7 (fasc. 1), pp. 325–340.
- Shumeyko A.A., Sotnik S.L. 2011. *Use of the agglomerative clustering for the automatic rubrication of texts*, System Technologies, 3(74), pp. 131–137.
- Simon J.C. 1986. *Patterns and Operators. The Foundations of Data Representation*, North Oxford Academic Publishers Ltd., London.
- Skalna I., Rębiasz B., Gawel B., Basiura B., Duda J., Opila J., Pelech-Pilichowski T. 2015. *Advances in Fuzzy Decision Making, Theory and Practice*, Springer International Publishing, DOI: 10.1007/978-3-319-26494-3.
- Smaldon J., Freitas A.A. 2006. *A new version of the ant-miner algorithm discovering unordered rule sets*, in: *Genetic and Evolutionary Computation Conference (GECCO'06)*, ACM Press, pp. 43–50.
- Smith J.F. 2002. *Data Mining for Fuzzy Decision Tree Structure with a Genetic Program*, in: Yin H., Allinson N., Freeman R., Keane J., Hubbard S. (ed.), *Intelligent Data Engineering and Automated Learning – IDEAL 2002*. 2002. Lecture Notes in Computer Science, vol. 2412, Springer, Berlin – Heidelberg, pp. 13–18.
- Sneath P.H.A., Sokal R.R. 1973. *Numerical Taxonomy*, W.H. Freeman, San Francisco.
- Sugeno M. 1985. *An introductory survey of fuzzy control*, *Information Sciences*, vol. 36(1), pp. 59–83.
- Sugeno M. 1985. *Industrial Applications of Fuzzy Control, Industrial Applications of Fuzzy Control*, Elsevier Science Inc., New York.
- Support Vector Machine/Wikipedia, [https://en.wikipedia.org/wiki/Support\\_vector\\_machine](https://en.wikipedia.org/wiki/Support_vector_machine) (access: 20.03.2018).
- Surowiecki J. 2005. *The Wisdom of the Crowds, Anchor; Reprint edition*, August 16, 2005.
- Swidan A., Sergey S., Dmitry B. 2013. *Using Genetic Algorithms for Solving the Comparison-Based Identification Problem of Multifactor Estimation Model*, Journal of Software Engineering and Applications, vol. 6, no. 7, pp. 349–353, DOI: 10.4236/jsea.2013.67044.
- Symeonidis P., Nanopoulos A., Papadopoulos A.N., Manolopoulos Y. 2006. *Collaborative filtering: Fallacies and insights in measuring similarity*, Universitaet Kassel, PKDD Workshop on Web Mining, Berlin.

- Takagi T., Sugeno M. 1985. *Fuzzy identification of systems and its applications to modeling and control*, IEEE Transactions on Systems, Man, and Cybernetics, vol. SMC-15, no. 1, pp. 116–132, DOI: 10.1109/TSMC.1985.6313399.
- Tanwar S., Ramani T., Tyagi S. 2018. *Dimensionality Reduction Using PCA and SVD in Big Data: A Comparative Case Study*, in: *Future Internet Technologies and Trends*, Springer International Publishing, pp. 116–125, DOI: 10.1007/978-3-319-73712-6\_12 (access: 20.02.2018).
- Terano T., Asai K., Sugeno M. (ed.). 1994. *Applied Fuzzy Systems*, Academic Press, Cambridge.
- Theodoridis S., Koutroumbas K. 2006. *Pattern Recognition*, Third Edition, Elsevier USA, Academic Press.
- Tian L.F., Kwok-Wai Cheung W. 2003. *Learning User Similarity and Rating Style for Collaborative Recommendation*, Information Retrieval.
- Tipping M.E., Bishop C.M. 1999. *Probabilistic Principal Component Analysis*, Journal of the Royal Statistical Society. Series B (Statistical Methodology), vol. 61, no. 3, pp. 611–622.
- Turksen I.B., Tian Y., Berg M. 1992. *A fuzzy expert system for a service centre of spare parts*, Expert Systems with Applications, vol. 5(3–4), pp. 447–464 DOI: 10.1016/0957-4174(92)90029-R (access: 20.03.2018).
- Vapnik V. 1998. *Statistical Learning Theory*, John Wiley & Sons, New York.
- Vapnik V. 2000. *The Nature of Statistical Learning Theory*, Series Information Science and Statistics, Springer, New York.
- Vapnik V., Chervonenkis A. 1964. *A class of algorithms for pattern recognition learning* [Об одном классе алгоритмов обучения распознаванию образов], *Avtomatika i Telemekhanika*, vol. 25(6), pp. 937–945, Moscow, Russia, <http://www.mathnet.ru/links/1546cee1510bf8d2b85d69a8179d52d8/at11678.pdf> (access: 18.04.2018).
- Vapnik V., Lerner A. 1963. *Pattern Recognition Using Generalized Portraits*, *Automation and Remote Control*, vol. 24, pp. 709–715.
- Vapnik, V., Chervonenkis, A. 1974. *Theory of Pattern Recognition*, Nauka, Moscow (in Russian); *German translation: Theorie der Zeichenerkennung*, Akademie Verlag, Berlin 1979, <http://www.citeulike.org/group/1938/article/1055538> (access: 18.04.2018).
- Veksler O. 2006. *Course „Pattern Recognition”*, The University of Western Ontario Department of Computer Science, [http://www.csd.uwo.ca/faculty/olga/Courses/Winter2006/CS434\\_654b/index.html](http://www.csd.uwo.ca/faculty/olga/Courses/Winter2006/CS434_654b/index.html) (access: 9.04.2018).
- von Luxburg U. 2007. *A tutorial on spectral clustering*, *Statistics and Computing*, vol. 17(4), pp. 395–416.



- von Luxburg U., Belkin M., Bousquet O. 2008. *Consistency of Spectral Clustering*, The Annals of Statistics, vol. 36, no. 2, Institute of Mathematical Statistics, pp. 555–586, DOI: 10.1214/009053607000000640.
- Weiss Y. 1999. *Segmentation using eigenvectors: a unifying view*, Proceedings of the Seventh IEEE International Conference on Computer Vision, pp. 975–982, IEEE, Kerkyra, Greece, 20–27 September, DOI: 10.1109/ICCV.1999.790354.
- Whitley D. 1994. *A genetic algorithm tutorial*, Statistics and Computing, vol. 4(2), pp. 65–85, DOI: 10.1007/BF00175354 (access: 20.03.2018).
- Whitley D. 2001. *An overview of evolutionary algorithms: practical issues and common pitfalls*, in: *Information and Software Technology*, vol. 43(14, 15), December 2001, pp. 817–831, DOI: 10.1016/S0950-5849(01)00188-4 (access: 20.03.2018).
- Widrow B., Hoff M.E. 1988. *Adaptive switching circuits*, in: Anderson J.A., Rosenfeld E. (ed.), *Neurocomputing: foundations of research*, MIT Press Cambridge, MA, pp. 123–134.
- Wolberg J. 2006. *Data Analysis Using the Method of Least Squares, Extracting the Most Information from Experiments*, Springer, Berlin – Heidelberg, DOI: 10.1007/3-540-31720-1.
- Wu C.F.J. 1983. *On the convergence properties of the EM algorithm*, The Annals of Statistics, vol. 11(1), pp. 95–103.
- Yager R.R. 1986. *An introduction to fuzzy set theory*, in: Karwowski W., Mitai A. (ed.), *Advances in Human Factors/Ergonomics*, vol. 6, pp. 29–39, DOI: 10.1016/B978-0-444-42723-6.50007-6 (access: 20.03.2018).
- Yager R.R., Ovchinnikov S., Tong R., Nguyen H. 1987. *Fuzzy Sets and Applications: Selected Papers by L.A. Zadeh*, John Wiley & Sons, New York.
- Yager R.R., Zadeh L.A. (ed.). 1992. *An Introduction to Fuzzy Logic Applications in Intelligent Systems*, Series The Springer International Series in Engineering and Computer Science, Springer, New York.
- Yang Y. 1999. *An Evaluation of Statistical Approaches to Text Categorization*, Information Retrieval, vol. 1, no. 1–2, pp. 69–90, DOI: 10.1023/A:1009982220290.
- Yang Y., Pedersen J.O. 1997. *A Comparative Study on Feature Selection in Text Categorization*, Proceedings of the Fourteenth International Conference on Machine Learning, July 08–12, 1997, San Francisco, CA, USA, Morgan Kaufmann Publishers Inc., pp. 412–420.
- Yang Y., Xin Liu. 1999. *A re-examination of text categorization methods*, Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval, pp. 42–49, 15–19 August 1999, Berkeley, CA, DOI: 10.1145/312624.312647.

- Zadeh L.A. 1973. *Outline of a new approach to the analysis of complex systems and decision processes*, IEEE Transactions on Systems, Man, and Cybernetics (SMC). vol. 3(1), pp. 28–44.
- Zadeh L.A. 1975. *The concept of a linguistic variable and its application to approximate reasoning*, Information Sciences, Part I: vol. 8, pp. 199–249; Part II: vol. 8, pp. 301–357; Part III: vol. 9, pp. 43–80.
- Zadeh L.A. 1996. *Fuzzy Sets, Fuzzy Logic, and Fuzzy Systems*, Advances in Fuzzy Systems – Applications and Theory, vol. 6, pp. 394–432.
- Zadeh L.A. 1965. *Fuzzy sets. Information and Control*, vol. 8(3), pp. 338–353.
- Zaki M.J., Meira W., Jr. 2014. *Data Mining and Analysis: Fundamental Concepts and Algorithms*, Cambridge University Press.
- Zelnik-Manor L., Perona P. 2004. *Self-Tuning Spectral Clustering*, Proceedings of the 18th Annual Conference on Neural Information Processing Systems (NIPS'04), <http://books.nips.cc/nips17.html>.
- Zhang H. 2004. *The Optimality of Naive Bayes*, Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference (FLAIRS 2004), AAAI Press, <http://www.cs.unb.ca/~hzhang/publications/FLAIRS04ZhangH.pdf> (access: 5.04.2018).
- Zheng F., Webb G.I. 2005. *A comparative study of semi-naive Bayes methods in classification learning*, Proceeding of the 4th Australasian Data Mining conference (AusDM05), pp. 141–156.
- Zheng F., Webb G.I. 2008. *Semi-naive Bayesian classification*, Journal of Machine Learning Research, vol. 87(1), pp. 93–125.

## **CD contents**

### **Each file contains programs from the sections of the book:**

Section 1: Least Squares Methods

Section 2: Principal Component Analysis

Section 4: Soft Computing in Data Handling

Section 5a: Hierarchical clustering

Section 5b: Nonhierarchical clustering

Section 9: Visualization of multidimensional data

### **Two folders with the object oriented application:**

Section 6.4.1: Example of sale of the Naive Bayes classifier 4.2.3

Simple GP: Example of GP form Section 4.2.3