UNIVERSITY OF ŽILINA

FACULTY OF MANAGEMENT SCIENCE AND INFORMATICS

DIPLOMA THESIS

FIELD OF STUDY: INFORMATION SYSTEMS – DATA PROCESSING

VOLODYMYR PONOMARENKO

Development of Expert System component for medical data analysis based on clustering

Supervisor: Prof., Dr. Paul Barach, MD, MPH

Tutor: prof. Ing. Vitaly Levashenko, PhD.

Registration number: 131/2017

Žilina, 2018

ŽILINSKÁ UNIVERZITA V ŽILINE, FAKULTA RIADENIA A INFORMATIKY.

ZADANIE TÉMY DIPLOMOVEJ PRÁCE.

Študijný program : Informačné systémy

Zameranie: Spracovanie dát

Meno a priezvisko
Volodymyr Ponomarenko

Osobné číslo
556989

Názov práce v slovenskom aj anglickom jazyku

Vývoj komponentu expertného systému pre analýzu lekárskych údajov s využitím zhlukovania

Development of Expert System component for medical data analysis based on clustering

Zadanie úlohy, ciele, pokyny pre vypracovanie (Ak je málo miesta, použite opačnú stranu)

Cieľ diplomovej práce:
The work's goal is development of the components of Expert System based on clustering algorithms

Obsah:
This work will be implemented in English
1. Definition of the clustering problem. Its place in medical data analysis
2. Analysis of existing algorithms of clustering
3. Design of a software for transformation from numerical into linguistic values
4. Implementation of the designed software in C++ programming language
5. The data analysis and accuracy evaluation

Meno a pracovisko vedúceho DP: Prof., Dr. Paul Barach, MD, MPH
Meno a pracovisko tútora DP: prof. Ing. Vitaly Levashenko, PhD.

vedúci katedry
(dátum a podpis

Zadanie zaregistrované dňa 24. 10. 2017 pod číslom 131/2017 podpis _____

Čestné vyhlásenie

Čestne prehlasujem, že som prácu vypracoval samostatne s využitím dostupnej literatúry a vlastných vedomostí. Všetky zdroje použité v diplomovej práci som uviedol v súlade s predpismi.

Honorable statement

I honestly declare that I have done the work independently using the available literature and my own knowledge. All the resources used in my thesis were stated in accordance with the regulations.

Žilina, _____ Volodymyr Ponomarenko

Poďakovanie

V prvom rade by som chcel poďakovať Pánu Bohu za rozum, zdravie a silu, ktoré mi dal pre napísanie tejto diplomovej práce, a tiež za mojich tútora a vedúceho, pod múdrym vedením ktorých táto práca bola úspešne dokončená.

Obrovská vďaka patri tútorovi diplomovej práce prof. Ing. Vitalymu Levashenkovi, PhD. za cenné rady, poskytnuté materiály, množstvo času venovaného konzultáciám, a za jeho trpezlivosť na všetkých etapách vypracovania tejto práce.

Chcem taktiež poďakovať všetkým kto sa modlil za mňa a veril že zvládnem nielen túto prácu ale aj celé inžinierske štúdium, hlavne mojej mame.

Thanks

First of all, I would like to thank the Lord God for the intellect, health and strength that He has given me to write my diploma thesis, as well as for my tutor and supervisor, under whose wise leadership this work was successfully completed.

Huge thanks belong to the tutor of the diploma thesis prof. Ing. Vitaly Levashenko, PhD. for valuable advices, provided materials, plenty of time spent on consultations, as well as for his patience during all stages of this work.

I also want to thank all who prayed for me and believed that I would be able to handle not only this work but also the whole study at master's degree level, especially I am grateful to my mom.

Abstrakt

PONOMARENKO, Volodymyr: Vývoj komponentu expertného systému pre analýzu lekárskych údajov s využitím zhlukovania. [Diplomová práca] – Žilinská univerzita v Žiline, Fakulta riadenia a informatiky, Katedra informatiky. – Vedúci: Prof., Dr. Paul Barach, MD, MPH. – Tútor: prof. Ing. Vitaly Levashenko, PhD. – Stupeň odbornej kvalifikácie: Inžinier v odbore Informatika. – Žilina: FRI ŽU v Žiline, 2018. – 112 s.

Diplomová práca sa zaoberá transformáciou numerických hodnôt na lingvistické v oblasti analýzy lekárskych údajov. Skúmajú sa rôzne algoritmy zhlukovania pre vykonanie tejto transformácie. Cieľom prace je vyvinúť softvérový komponent expertného systému založený na zhlukovaní.

Kľúčové slová: algoritmy zhlukovania, analýza lekárskych údajov, fuzzifikácia, fuzzy zhlukovanie, indexy platnosti klastrov.

Abstract

PONOMARENKO, Volodymyr: Development of Expert System component for medical data analysis based on clustering. [Diploma thesis] – University of Žilina, Faculty of Management Science and Informatics, Department of informatics. – Supervisor: Prof., Dr. Paul Barach, MD, MPH. – Tutor: prof. Ing. Vitaly Levashenko, PhD. – Qualification level: Master of Computer Science. – Žilina: FRI ŽU in Žilina, 2018. – 112 p.

The diploma thesis is devoted to the transformation of numerical values into linguistic values in medical data analysis. Different clustering algorithms are considered to perform the transformation. The goal of the work is to develop a software component of the Expert System based on clustering.

Key words: clustering algorithms, cluster validity indices, fuzzification, fuzzy clustering, medical data analysis.

CONTENTS

List of Figures 8
List of Tables 12
List of Abbreviations 13
INTRODUCTION 14
CHAPTER 1. THEORETICAL ASPECTS OF CLUSTERING 16
1.1. Fundamentals of clustering 16
1.1.1. Definition of the clustering problem 16
1.1.2. Data types 19
1.1.3. Similarity Measuring 20
1.2. Clustering in medical data analysis 24
1.3. Fuzzy clustering 26
CHAPTER 2. CLUSTERING ALGORITHMS 30
2.1. Fundamentals of clustering algorithms 30
2.2. Classical fuzzy clustering algorithms 34
2.2.1. Fuzzy c-Means Clustering 34
2.2.2. Gustafson-Kessel Clustering Algorithm 35
2.2.3. Gath-Geva Clustering Algorithm 37
2.2.4. Determining the optimal number of clusters 39
2.3. Multi-Interval Discretization 41
2.4. Fuzzy Entropy Based Fuzzy Classifier 46
2.5. Fuzzy Information Density Based Fuzzy Classifier 51
CHAPTER 3. A SOFTWARE TOOL FOR DATA ANALYSIS BASED ON CLUSTERING 54
3.1. Design of the Fuzzy Clustering Tool 54
3.2. Core module implementation 56
3.3. Graphical user interface implementation 72
CHAPTER 4. EXPERIMENTAL STUDY WITH THE IMPLEMENTED SOLUTION 77
4.1. Fuzzy clustering accuracy evaluation 77
4.2. Comparison of the fuzzy clustering algorithms on medical data 79

4.2.1. Pima Indians Diabetes 79
4.2.2. Heart Disease 84
4.2.3. Breast Cancer Wisconsin 88
4.2.4. Indian Liver Patient Records 94
4.2.5. Chronic Kidney Disease 98
CONCLUSION 104
BIBLIOGRAPHY 106
List of Appendices 109
APPENDICES 110
Appendix 1: UML Class diagram of the Fuzzy Clustering Tool 111
Appendix 2: Contents of the CD 112

List of Figures

Figure 1.1. Data mining tasks 16
Figure 1.2. Simple example of clustering 18
Figure 1.3. Diagram of data types 19
Figure 1.4. Examples of distance functions – three-dimensional and contour plots 23
Figure 1.5. A set of two-dimensional patterns, that contains three clusters. The patterns, which can belong to more than one cluster are pointed to by the arrows 26
Figure 1.6. Classification of approaches for cluster analysis 27
Figure 2.1. Phases of a clustering algorithm 30
Figure 2.2. Criterion function impact on a clustering quality 31
Figure 2.3. Example of the MID clustering algorithm, given by Popel 44
Figure 2.4. The fuzzy membership function with boundaries and centers of clusters 45
Figure 2.5. Example of a distribution of 3 classes of objects with corresponding membership functions 49
Figure 3.1. The Use Case diagram of the Fuzzy Clustering Tool 55
Figure 3.2. A structure of the Fuzzy Clustering Tool implementation as a “solution” in Visual Studio 2017 56
Figure 3.3. A Solver class view 57
Figure 3.4. Enumerators in the core module 59
Figure 3.5. A Dataset class view 60
Figure 3.6. A DatasetDescription class view 61
Figure 3.7. An Attribute class view 62
Figure 3.8. An AttributeHeader class view 63
Figure 3.9. A NumericAttribute class view 64
Figure 3.10. A NominalAttribute class view 65
Figure 3.11. An AttributeData abstract class with derived classes 67
Figure 3.12. A Fuzzification class view 68
Figure 3.13. A FuzzyClusteringAlgorithm abstract class view 68
Figure 3.14. Classes, that implement FCM and its modifications 69
Figure 3.15. A MID class view 70
Figure 3.16. Classes, implementing FEBFC algorithm and its modification FIDBFC 71

Figure 3.17. A FuzzyClustersImporter class view 72
Figure 3.18. The main window of the Fuzzy Clustering Tool 73
Figure 3.19. The *Dataset* menu 74
Figure 3.20. The *Fuzzy* menu 74
Figure 3.21. "Current dataset" section of the main window of the tool 75
Figure 3.22. "Attributes" section of the main window of the tool 75
Figure 3.23. "Selected attribute" section of the main window of the tool 75
Figure 3.24. "Clusters" section of the main window of the tool 76
Figure 3.25. "Visualization" section of the main window of the tool 76
Figure 4.1. Membership functions of attributes $\mathbf{A_1}$ – $\mathbf{A_8}$ obtained using the FCM algorithm with Pairing Frequency index for getting the optimal number of clusters 80
Figure 4.2. Membership functions of attributes $\mathbf{A_1}$ – $\mathbf{A_8}$ obtained using the GK algorithm with Pairing Frequency index for getting the optimal number of clusters 80
Figure 4.3. Membership functions of attributes $\mathbf{A_1}$ – $\mathbf{A_8}$ obtained using the GG algorithm with Pairing Frequency index for getting the optimal number of clusters 81
Figure 4.4. Membership functions of attributes $\mathbf{A_1}$ – $\mathbf{A_8}$ obtained using the MID algorithm 81
Figure 4.5. Membership functions of attributes $\mathbf{A_1}$ – $\mathbf{A_8}$ obtained using the FEBFC algorithm 82
Figure 4.6. Membership functions of attributes $\mathbf{A_1}$ – $\mathbf{A_8}$ obtained using the FIDBFC algorithm 82
Figure 4.7. Membership functions of attributes $\mathbf{A_1}$, $\mathbf{A_4}$, $\mathbf{A_5}$, $\mathbf{A_8}$ and $\mathbf{A_{10}}$ obtained using the FCM algorithm with Pairing Frequency index for getting the optimal number of clusters 85
Figure 4.8. Membership functions of attributes $\mathbf{A_1}$, $\mathbf{A_4}$, $\mathbf{A_5}$, $\mathbf{A_8}$ and $\mathbf{A_{10}}$ obtained using the GK algorithm with Pairing Frequency index for getting the optimal number of clusters 86
Figure 4.9. Membership functions of attributes $\mathbf{A_1}$, $\mathbf{A_4}$, $\mathbf{A_5}$, $\mathbf{A_8}$ and $\mathbf{A_{10}}$ obtained using the GG algorithm with Pairing Frequency index for getting the optimal number of clusters 86
Figure 4.10. Membership functions of attributes $\mathbf{A_1}$, $\mathbf{A_4}$, $\mathbf{A_5}$, $\mathbf{A_8}$ and $\mathbf{A_{10}}$ obtained using the MID algorithm 86

Figure 4.11. Membership functions of attributes **A1**, **A4**, **A5**, **A8** and **A10** obtained using the FEBFC algorithm 87
Figure 4.12. Membership functions of attributes **A1**, **A4**, **A5**, **A8** and **A10** obtained using the FIDBFC algorithm..... 87
Figure 4.13. Membership functions of attributes **A1** – **A10** obtained using the FCM algorithm with Pairing Frequency index for getting the optimal number of clusters 90
Figure 4.14. Membership functions of attributes **A1** – **A10** obtained using the GK algorithm with Pairing Frequency index for getting the optimal number of clusters 90
Figure 4.15. Membership functions of attributes **A1** – **A10** obtained using the GG algorithm with Pairing Frequency index for getting the optimal number of clusters 91
Figure 4.16. Membership functions of attributes **A1** – **A10** obtained using the MID algorithm..... 91
Figure 4.17. Membership functions of attributes **A1** – **A10** obtained using the FEBFC algorithm..... 92
Figure 4.18. Membership functions of attributes **A1** – **A10** obtained using the FIDBFC algorithm..... 92
Figure 4.19. Membership functions of attributes **A1**, **A3** – **A10** obtained using the FCM algorithm with Pairing Frequency index for getting the optimal number of clusters 95
Figure 4.20. Membership functions of attributes **A1**, **A3** – **A10** obtained using the GK algorithm with Pairing Frequency index for getting the optimal number of clusters 95
Figure 4.21. Membership functions of attributes **A1**, **A3** – **A10** obtained using the GK algorithm with Pairing Frequency index for getting the optimal number of clusters 96
Figure 4.22. Membership functions of attributes **A1**, **A3** – **A10** obtained using the MID algorithm..... 96
Figure 4.23. Membership functions of attributes **A1**, **A3** – **A10** obtained using the FEBFC algorithm..... 97
Figure 4.24. Membership functions of attributes **A1**, **A3** – **A10** obtained using the FIDBFC algorithm..... 97

Figure 4.25. Membership functions of attributes **A1**, **A2**, **A10** – **A18** obtained using the FCM algorithm with Pairing Frequency index for getting the optimal number of clusters 100
Figure 4.26. Membership functions of attributes **A1**, **A2**, **A10** – **A18** obtained using the GK algorithm with Pairing Frequency index for getting the optimal number of clusters 100
Figure 4.27. Membership functions of attributes **A1**, **A2**, **A10** – **A18** obtained using the GG algorithm with Pairing Frequency index for getting the optimal number of clusters 101
Figure 4.28. Membership functions of attributes **A1**, **A2**, **A10** – **A18** obtained using the MID algorithm 101
Figure 4.29. Membership functions of attributes **A1**, **A2**, **A10** – **A18** obtained using the FEBFC algorithm 102
Figure 4.30. Membership functions of attributes **A1**, **A2**, **A10** – **A18** obtained using the FIDBFC algorithm 102

List of Tables

Table 1.1. Comparison of different definitions of the term "clustering" by key characteristics 18
Table 1.2. The most common indices of similarity between patterns **x** and **y** depending on values of a, b, c and d entries 22
Table 1.3. The most common distance functions between patterns **x** and **y** 22
Table 2.1. Optimality criterions of fuzzy clustering (Cluster Validity Indices) 32
Table 4.1. Attributes of the Pima Indians Diabetes data set: $\mathbf{A_1}$-$\mathbf{A_8}$ are input attributes and **C** is an output attribute 79
Table 4.2. Clustering Accuracy Indices calculated for the fuzzification performed on the Pima Indians Diabetes Dataset 83
Table 4.3. Attributes of the Cleveland Heart Disease data set: $\mathbf{A_1}$-$\mathbf{A_{13}}$ are input attributes and **C** is an output attribute 84
Table 4.4. Clustering Accuracy Indices calculated for the fuzzification performed on the Cleveland Heart Disease data set 88
Table 4.5. Attributes of the Breast Cancer Wisconsin data set: $\mathbf{A_1}$-$\mathbf{A_{10}}$ are input attributes and **C** is an output attribute 89
Table 4.6. Clustering Accuracy Indices calculated for the fuzzification performed on the Breast Cancer Wisconsin data set 93
Table 4.7. Attributes of the Liver Patient Records data set: $\mathbf{A_1}$-$\mathbf{A_{10}}$ are input attributes and **C** is an output attribute 94
Table 4.8. Clustering Accuracy Indices calculated for the fuzzification performed on the Indian Liver Patient Records data set 98
Table 4.9. Attributes of the Chronic Kidney Disease data set: $\mathbf{A_1}$-$\mathbf{A_{24}}$ are input attributes and **C** is an output attribute 99
Table 4.10. Clustering Accuracy Indices calculated for the fuzzification performed on the Chronic Kidney Disease data set 103

List of Abbreviations

CVI	Cluster Validity Index
EU	European Union
FCM	Fuzzy c-Means
FEBFC	Fuzzy Entropy Based Fuzzy Classifier
FIDBFC	Fuzzy Information Density Based Fuzzy Classifier
FMLE	Fuzzy Maximum Likelihood Estimation
GG	Gath-Geva
GK	Gustafson-Kessel
GUI	Graphical User Interface
HIPAA	Health Insurance Portability and Accountability Act
IDE	Integrated Development Environment
ISODATA	Iterative Self-Organizing Data Analysis Technique
MID	Multi-Interval Discretization
MVC	Model-View-Controller
NMI	Normalized Mutual Information
PBMF	Pakhira-Bandyopadhyay-Maulik Fuzzy

INTRODUCTION

During the last few decades, the face of the modern world was qualitatively changed by information technologies. Various technical means and information delivery channels, based on progressive information and communication systems, have revolutionized human life, that nowadays is inseparably associated with huge flows of data. These data are involved in all spheres of human activity – social, economic, political, spiritual. However, the information contained in such data can be whether very valuable or completely useless. Extracting various kinds of useful information from the data sets is one of the main tasks of the modern science called data mining.

Data mining proposes methods, that today are widely used in healthcare, which is in one of the fundamental fields of the social sphere of modern life. Such popularity of the data mining methods was formed mainly due to the rapid development of medical devices and therapy technologies, that allow to produce and to store large amount of data. Producing of new data is mostly achieved in a process of providing medical services. For example, imagine a patient who came to a polyclinic to examine his digestive system. A medical worker, using a special probe or ultrasound device, measures necessary medical indicators and records them in a medical report. Obtained in a such way medical data are usually persisted in some database for possible using in future. Therefore, storing medical data is achieved through the use of various database systems.

The information contained in medical data is extremely important for solving diagnostic, therapeutic, statistical, administrative and other tasks in the field of medicine, e.g. determination of a correct treatment, definition of a patient's group of risk and prevention of diseases. Solving of these tasks has a huge impact on a quality of medical services, life expectancy, mortality and time of illnesses of population.

In the field of medicine, both numeric (continuous) and nominal (linguistic) types of data are used. The numeric data type is used for representing value of a continuous medical indicator, e.g. age of a patient, body mass index, resting blood pressure, albumin and globulin ratio. Variables of the nominal data type usually keep a name of some state of a categorical medical indicator, e.g. a patient's appetite can be good or poor, a tumor can be malignant or benign. In medical data analysis obtaining a continuous value is often not enough informative to make a conclusion about patient's state for determining necessary

treatment, so the medical worker has to associate this value with an appropriate nominal value. This is a typical situation when a transformation from numeric into linguistic values comes into play. The numeric values of an attribute obtained as a result of the transformation can be relatively easy converted into fuzzy data specified by a membership function. The whole process of obtaining fuzzy values from numeric values is called fuzzification.

Fuzzy medical data, obtained as a result of fuzzification, are very valuable. They can be used, in particular, for increasing the healthcare system reliability through the reducing potential medical failures, that is discussed in papers [1]-[2].

The process of transformation from numeric into linguistic values, that is based on cluster analysis, is the subject of study in this diploma thesis. The Expert System for medical data analysis is the object of study. The work's goal is development of a software, that should be able to transform values of any numeric attribute of the medical data set into fuzzy values. It must be possible to use this software, based on clustering algorithms, as a computational component of Expert System for medical data analysis. The information, containing in the resulting fuzzy data sets, is very valuable, because can be used in constructing predictive components of the Expert System, e.g. decision trees.

The diploma thesis consists of four chapters. In Chapter 1 theoretical aspects of clustering problem is considered, that include describing a place of cluster analysis in the data mining science and its usage for medical data analysis, definition of the term “clustering” and formulation the clustering goals, familiarizing with different data types and measures of similarity and dissimilarity, and explanation differences between hard clustering and fuzzy clustering.

Chapter 2 is devoted to describing Cluster Validity Indices and following clustering algorithms: Fuzzy c-Means, Gustafson-Kessel algorithm, Gath-Geva algorithm, Multi-Interval Discretization and Fuzzy Entropy Based Fuzzy Classifier. Additionally, this chapter includes a modification of the Fuzzy Entropy Based Fuzzy Classifier, proposed by the author of this thesis.

Chapter 3 describes a design and implementation of a software tool for transformation numeric values into fuzzy values based on clustering. This software is called Fuzzy Clustering Tool. Chapter 4 contains experimental study on medical data sets. In this study the implemented in the tool clustering algorithms are evaluated and compared using the Clustering Accuracy Indices.

CHAPTER 1. THEORETICAL ASPECTS OF CLUSTERING

1.1. Fundamentals of clustering

1.1.1. Definition of the clustering problem

According to L. Rokach and O. Maimon, data mining can be defined as "the science and technology of exploring data in order to discover previously unknown patterns" [3]. One of the fundamental problems in data mining is clustering (also called cluster analysis). Its role in data mining can be illustrated with the schema on Figure 1.1. Clustering belongs to indirect or unsupervised data mining, since we do not know anything about clusters we are looking for. Moreover, the purpose of cluster analysis is to determine the set of clusters for dividing an initial data set into.

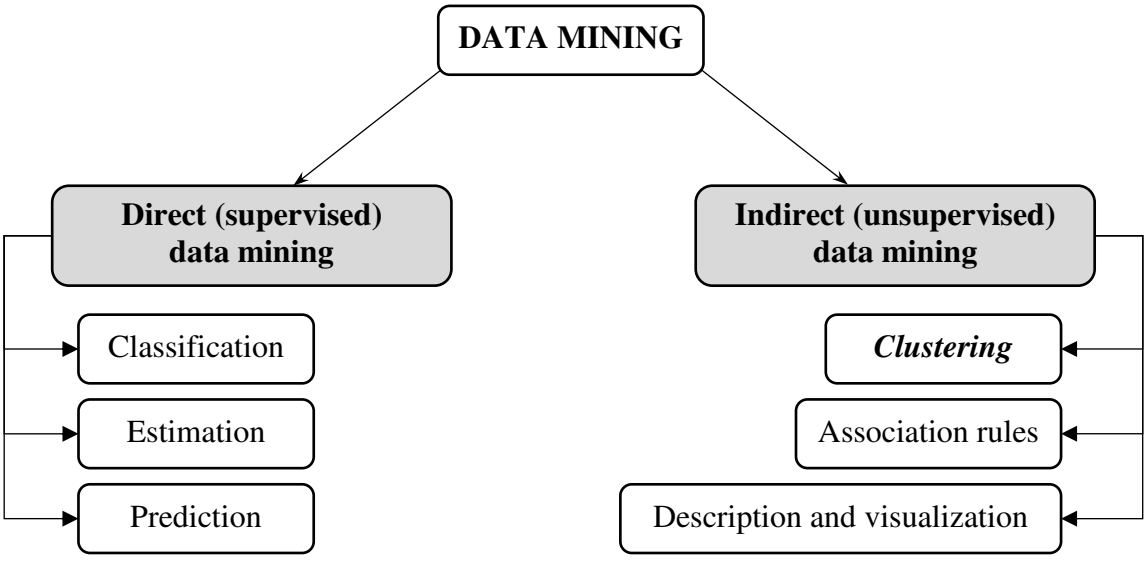


Figure 1.1. Data mining tasks [4].

The history of clustering goes back to Aristotle's attempt to classify living organisms [8]. Nowadays, cluster analysis has a wide range of fields of application, e.g. text mining, market research, business failure categorization, grouping of shopping items, credit

evaluation, social network analysis, human gene analysis, anomaly detection, crime analysis, image processing as well as medical data analysis.

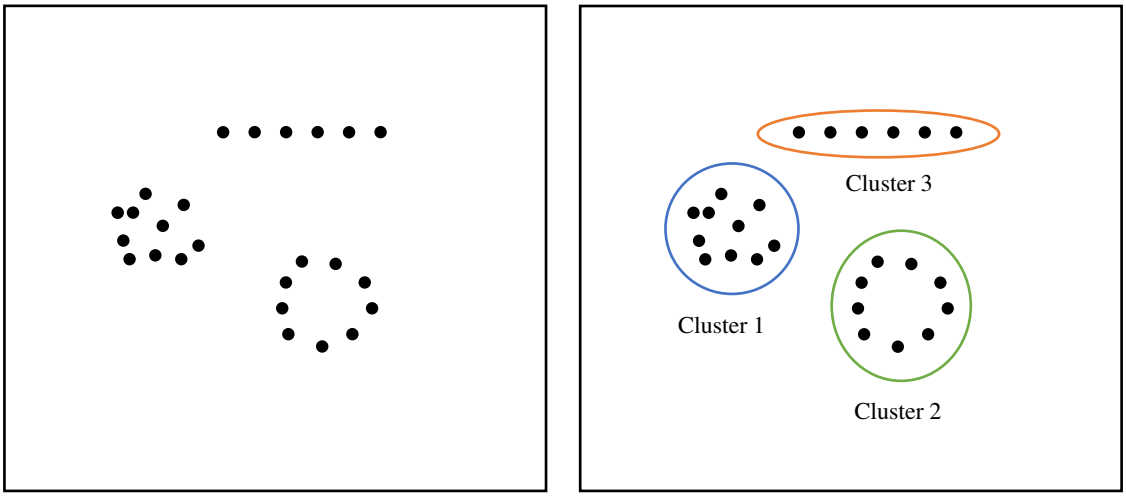
For moving deeper in this diploma thesis, we need to formalize, what the clustering is. L. Rokach and O. Maimon note about what clustering actually does, that it "groups the data instances into subsets in such a manner that similar instances are grouped together; different instances belong to different groups" [3]. According to G. Gan, M. Chaoqun and W. Jianhong, clustering "is a method of creating groups of objects, or clusters, in such a way that objects in one cluster are very similar and objects in different clusters are quite distinct" [4]. J. Abonyi and B. Feil define clustering as "the classification of similar objects into different groups, or more precisely, the partitioning of a data set into subsets (clusters), so that the data in each subset (ideally) share some common trait – often proximity according to some defined distance measure" [5]. Rui Xu and Donald C. Wunsch II explain the term of "clustering" as an "unsupervised classification" the goal of which is "to separate a finite, unlabeled data set into a finite and discrete set of "natural", hidden data structures, rather than to provide an accurate characterization of unobserved samples generated from the same probability distribution" [6]. Another definition of the term of "clustering" is based on understanding clustering problem as "categorizing or segmenting observations into groups or clusters such that each cluster is as homogeneous as possible", where the mentioned groups (clusters) "are usually unknown to or not predetermined by data miners" [7].

Analysis of the definitions of the term "clustering" in the previous paragraph leads us to extract key characteristics of clustering. Comparison of different definitions of the term is shown on Table 1.1. This is a process of partitioning a data set into groups (clusters), finite number of clusters, clustering is a type of classification, unsupervised learning, similarity is based on distance measure. We cannot agree with definitions, that explain clustering as some type of classification, because clustering belongs to indirect data mining while classification belongs to direct one. Unfortunately, none of the analyzed definitions include all mentioned characteristics except "clustering is a type of classification". This is the reason to make our own definition of clustering, which would improve the situation. We can define the clustering as the unsupervised learning method of partitioning a data set into a finite number of discrete groups (clusters) such that each group consists of similar objects according to some defined distance measure.

Table 1.1. Comparison of different definitions of the term “clustering” by key characteristics

Characteristic	[3]	[4]	[5]	[6]	[7]	Own definition
Partitioning a data set into groups (clusters)	+	+	+	+	+	+
Finite number of clusters	–	–	–	+	–	+
Discrete clusters	–	–	–	+	–	+
Clustering is a type of classification	–	–	+	+	+	–
Unsupervised learning	–	–	–	+	+	+
Similarity is based on distance measure	–	–	+	–	–	+

The cluster analysis principle can be demonstrated graphically in Figure 1.2, where (a) – the initial data set of objects (patterns, entities, instances) and (b) – a set of clusters into which the data set can be divided.



(a) (b)

Figure 1.2. Simple example of clustering.

It was mentioned above, that clustering belongs to unsupervised learning. The reason for this lays in a lack of prior information in the initial data sets. For example, disease diagnosis and treatment in clinics, where several unknown subtypes for each type of disease may exist, can be considered. Even if they all have similar morphological appearances, responding to the same therapy may differ. Clustering with gene expression data that measure the activities of genes can be used to solve this problem. It provides a promising

method to uncover the previously unknown subtypes of disease that will allow to determine the most appropriate therapies [6].

Clustering has four major goals, which cover all major aspects of analysis:

- development of a classification;
- investigation of useful conceptual schemes for grouping entities;
- hypothesis generation through data exploration;
- hypothesis testing or the attempt to determine if types defined through other procedures are in fact present in a data set [6].

Cluster analysis is realized through clustering algorithms, which will be discussed in Chapter 2.

1.1.2. Data types

Understanding of data types is obligatory for interpreting the results of cluster analysis, because each data type has its own field of usage depending on information that an attribute can contains. The most commonly used data types in clustering are discrete and continuous types. The discrete data type aggregates nominal and binary (symmetrical as well as asymmetrical) types (see Figure 1.3).

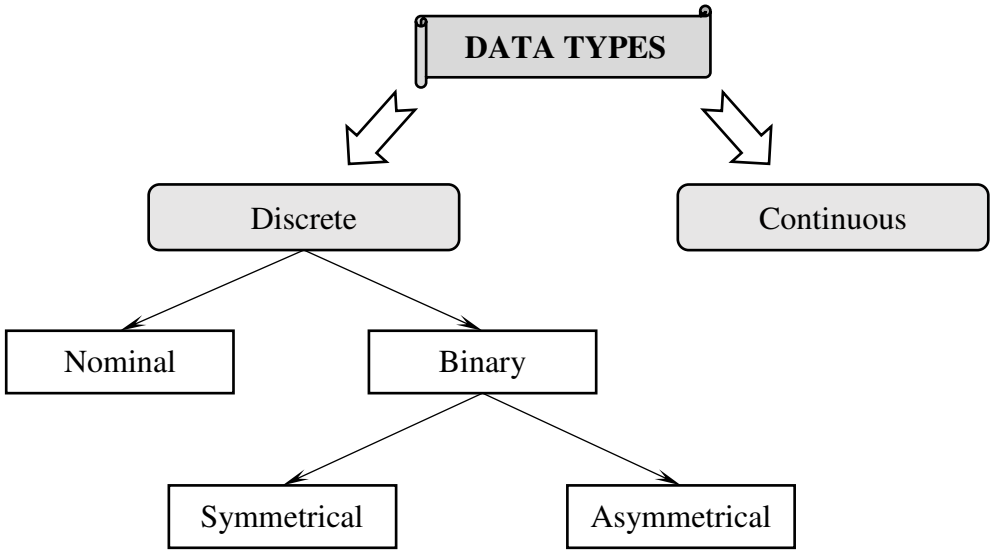


Figure 1.3. Diagram of data types [4].

The discrete data type is used for representing value of a discrete attribute. Number of possible values of such variable is finite. Similarly, the continuous data type is used for representing value of a continuous (numeric) attribute that can assume any value in $\mathbb{R}$.

A nominal (categorical) variable usually keeps a name of some object's state. For example, notebook manufacturer name is the nominal attribute of the notebook object. The categorical attribute assumes a finite number of values (as categorical data type belongs to discrete type), but nothing can be said about closeness of such values.

Binary variables keep one of two possible values (e.g. “true” or “false”). Depending on information importance of each value, a binary attribute can be symmetric or asymmetric. In a symmetric binary variable the importance of both possible values is the same (e.g. “black” or “white”), while in an asymmetric variable one value is more important than the other (e.g. “yes” stands for the presence and “no” stands for the absence of a certain attribute) [4].

1.1.3. Similarity Measuring

In Section 1.1.2 it was mentioned, that the result of clustering is a set of partitions (clusters) with following characteristics:

- patterns, that belongs to the same partition are homogeneous (as similar as possible);
- patterns, that belongs to different partitions are heterogeneous (as different as possible) [9].

Since we use a term of similarity here, it should be specified what the similarity is and how it can be measured. In the literature of clustering, under the concept of similarity is understood a presence of similar attributes in analyzed objects. The greater the similarity measure, the more similar two objects are [4].

For any objects $\mathbf{x}$ and $\mathbf{y}$ the similarity measure is a function, that satisfies following conditions:

- Positivity: $0 \leq s(\mathbf{x}, \mathbf{y}) \leq 1$ (1.1)
- Symmetry: $s(\mathbf{x}, \mathbf{y}) = s(\mathbf{y}, \mathbf{x})$ (1.2)

The function is called a similarity metric if it also satisfies the following additional conditions for any objects **x**, **y** and **z** [6]:

- $s(\mathbf{x},\mathbf{y}) \times s(\mathbf{y},\mathbf{z}) \leq \big(s(\mathbf{x},\mathbf{y}) + s(\mathbf{y},\mathbf{z})\big) \times s(\mathbf{x},\mathbf{z})$ (1.3)
- $s(\mathbf{x},\mathbf{x}) = 1$ (1.4)

Mostly the similarity measure is used with binary variables. Consider two vectors of binary data **x** and **y** (patterns, that have binary attributes). Let designate a number of occurrences when x_i and y_i are both equal to 1 as "a"; a number of occurrences when $x_i = 0$ and $y_i = 1$ as "b"; a number of occurrences when $x_i = 1$ and $y_i = 0$ as "c"; a number of occurrences when x_i and y_i are both equal to 0 as "d". Then these four numbers can be organized into a 2 by 2 co-occurrence matrix (contingency table) that visualizes how "close" these two objects are to each other [10]:

		x_i	
		1	0
y_i	1	a	b
	0	c	d

(1.5)

The highest similarity is in case of nondiagonal elements of matrix (1.5) are equal to zero. For measuring similarity depending on values of a, b, c and d entries, indices of similarity can be used. The most common indices are listed in Table 1.2.

For measuring similarity between continuous variables, the most appropriate measure is distance. To be more precise, distance is a measure of dissimilarity and can be defined by the following relation:

$$d(\mathbf{x},\mathbf{y}) = 1 - s(\mathbf{x},\mathbf{y}) \tag{1.10}$$

For any objects **x**, **y** and **z**, distance is a function, that satisfies following conditions:

- Positivity: $d(\mathbf{x},\mathbf{y}) \geq 0$ (1.11)
- Reflexivity: $d(\mathbf{x},\mathbf{x}) = 0$ (1.12)
- Symmetry: $d(\mathbf{x},\mathbf{y}) = d(\mathbf{y},\mathbf{x})$ (1.13)
- Triangle inequality: $d(\mathbf{x},\mathbf{y}) + d(\mathbf{y},\mathbf{z}) \geq d(\mathbf{x},\mathbf{z})$ (1.14)

Table 1.2. The most common indices of similarity between patterns **x** and **y** depending on values of a, b, c and d entries (a – number of occurrences when x_i and y_i are both equal to 1; b – number of occurrences when $x_i = 0$ and $y_i = 1$; c – number of occurrences when $x_i = 1$ and $y_i = 0$; d – number of occurrences when x_i and y_i are both equal to 0)

Index of similarity	Formula
Russel and Rao index	$s(\mathbf{x},\mathbf{y}) = \frac{a}{a+b+c+d}$ (1.6)
Sokal index	$s(\mathbf{x},\mathbf{y}) = \frac{a+d}{a+b+c+d}$ (1.7)
Jaccard index	$s(\mathbf{x},\mathbf{y}) = \frac{a}{a+b+c}$ (1.8)
Czekanowski index	$s(\mathbf{x},\mathbf{y}) = \frac{2a}{2a+b+c}$ (1.9)

Calculating distance gives us enough information to conclude how close objects are and depending on this closeness decide about their belonging to the appropriate clusters. The most common approaches for distance measuring are listed in Table 1.3.

Table 1.3. The most common distance functions between patterns **x** and **y** [10]

Distance function	Formula
Euclidean distance	$d(\mathbf{x},\mathbf{y}) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$ (1.15)
Hamming distance	$d(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{n}\lvert x_i - y_i\rvert$ (1.16)
Tchebyschev distance	$d(\mathbf{x},\mathbf{y}) = \max_{i=1,2,\dots,n}\lvert x_i - y_i\rvert$ (1.17)
Minkowski distance	$d(\mathbf{x},\mathbf{y}) = \sqrt[p]{\sum_{i=1}^{n}(x_i - y_i)^p}, p > 0$ (1.18)
Canberra distance	$d(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{n}\frac{\lvert x_i - y_i\rvert}{x_i + y_i}, x_i > 0, y_i > 0$ (1.19)
Mahalanobis distance	$d(\mathbf{x},\mathbf{y}) = \mathbf{x}^T A^{-1}\mathbf{y}$, where A is a positive definite matrix (1.20)

Approaches listed above in Table 1.3 assumes different data representation, that can be found in their geometric constructs, which are the contours of the constant distances between any two objects [10]. Examples of the geometric constructs for Euclidean, Hamming and Tchebyschev distances are shown in Figure 1.4.

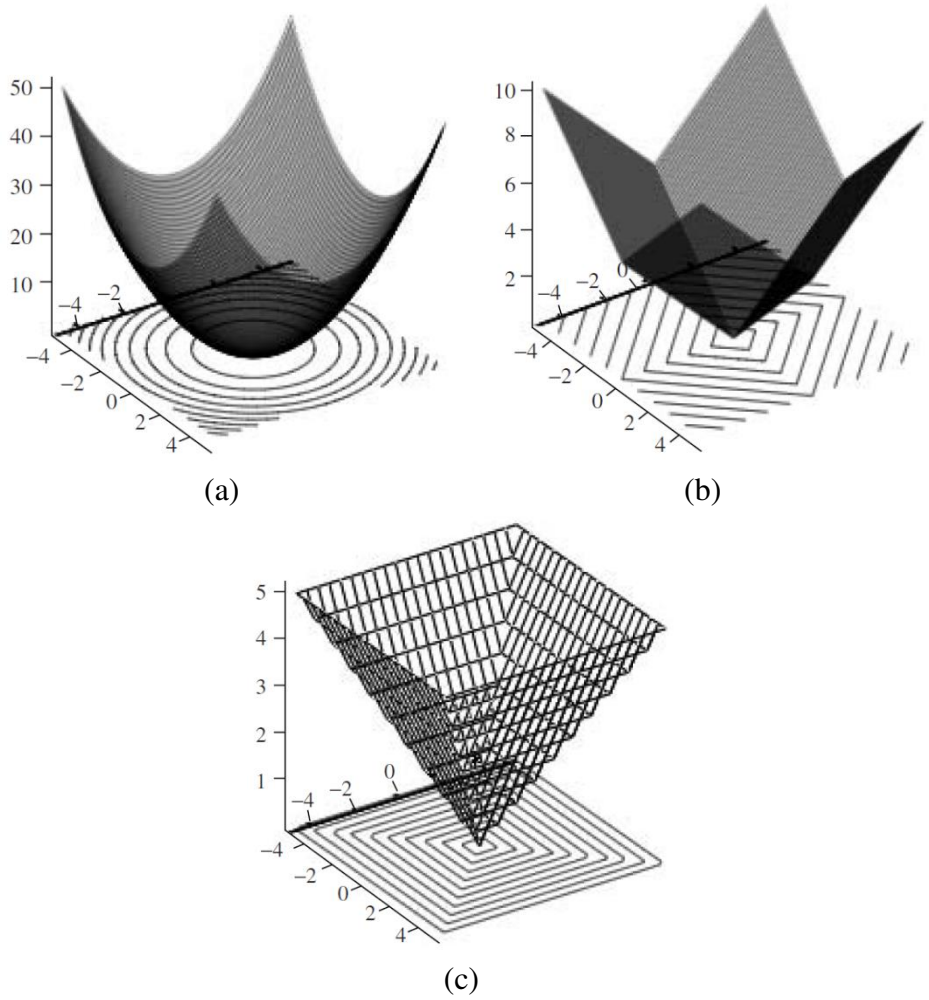


Figure 1.4. Examples of distance functions – three-dimensional and contour plots: (a) Euclidean, (b) Hamming (city block), (c) Tchebyschev [10]

1.2. Clustering in medical data analysis

Healthcare today is in one of the fundamental fields of the social sphere of human life. Over the last few decades, medical devices and therapy technologies has developed rapidly and nowadays they allow to produce and to store large amount of data. Producing of new data is mainly achieved in a process of providing medical services. For example, imagine a patient who came to a polyclinic to examine his digestive system. A medical worker, using a special probe or ultrasound device, measures necessary medical indicators and records them in a medical report. Obtained in a such way medical data are usually persisted in some database for possible using in future. Therefore, storing medical data is achieved through the use of various database systems.

All biologically active processes in a human organism are related with producing of various signal – electromagnetic, sonic, and mechanical. Mentioned above medical indicators of the human condition can be considered as signals (for example, growth, body weight, composition of blood and other biological fluids). Any objective or subjective sign of a disease is also a signal (e.g. patient complaints, fever, jaundice, results of physical survey) [11].

Some changes of properties of the latter can occur, as a result of interaction of biological signals, generated by the human body, with physical bodies (detectors). These changes are then registered by special medical devices as signals. In computer science such registered signals are known as medical data [11].

The information contained in medical data is extremely important for solving diagnostic, therapeutic, statistical, administrative and other tasks in the field of medicine, e.g. determination of a correct treatment, definition of a patient's group of risk and prevention of diseases. Solving of these tasks has a huge impact on a quality of medical services, life expectancy, mortality and time of illnesses of population.

In the field of medicine, both numeric (continuous) and nominal (linguistic) types of data are used. The numeric data type is used for representing value of a continuous medical indicator, e.g. age of a patient, body mass index, resting blood pressure, albumin and globulin ratio. Variables of the nominal data type usually keep a name of some state of a categorical medical indicator, e.g. a patient's appetite can be good or poor, a tumor can be malignant or benign. In medical data analysis obtaining a continuous value is often not enough informative to make a conclusion about patient's state for determining necessary

treatment, so the medical worker has to associate this value with an appropriate nominal value. This is a typical situation when clustering of medical data, that results in transformation from numeric into linguistic values, comes into play.

The main problem of medical data analysis is providing a decision support system for medical predictions, that would be efficient in determining of disease diagnosis and following treatment of a patient. Nowadays, the most promising methods of solving the mentioned problem are fuzzy logic, neural network, and machine learning algorithms [12]. All of them are based on clustering, that has become probably the most widely used data mining technique for medical data [13].

Numerous specific examples of the use of clustering in medical data analysis can be found in the literature. For example, clustering is used for identifying groups of genes that have similar biological functions, cancer class discovery and prediction, characterizing of psychiatric patients on the basis of clusters of symptoms, identifying medical patient groups that need specific targeted interventions, analyzing various signals and their relationships with particular diseases or symptoms [13, 14].

Medical data often contains confidential information relating to patients. The use of this information is governed by regulations, such as the EU Data Protection Directive (in the EU) or the 1996 Health Insurance Portability and Accountability Act (HIPAA) (in the United States). Therefore, before starting analysis, the data must be transformed so that the patient's personal information is not individually identifiable, that is, record should not contain sufficient data to identify the person associated with the record. Thus, cluster analysis is allowed only on the following types of medical data: anonymous data (data collected without patient-identification information), anonymized data (data collected with patient-identification information which is removed later), or de-identified data (data with patient-identification information encoded or encrypted) [13].

1.3. Fuzzy clustering

Initially, the cluster analysis was based on the classical set theory and implied an unambiguous separation of the entire data set into mutually exclusive clusters. It means that each object has a clearly defined membership in one and only one of the clusters based on a certain similarity or dissimilarity measure (discussed earlier in Section 1.1.3) [15]. But in practice in many cases an object can belong to several strictly defined clusters. For example, consider a set of two-dimensional patterns that contains three clusters (see Figure 1.5).

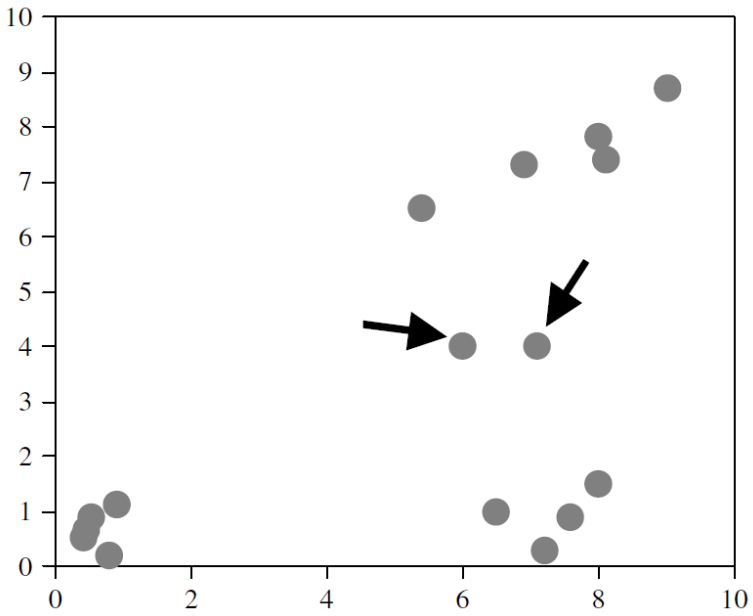


Figure 1.5. A set of two-dimensional patterns, that contains three clusters. The patterns, which can belong to more than one cluster are pointed to by the arrows [10].

As can be seen in Figure 1.5, although most patterns have an obvious belonging to only one of the clusters, two of them are on the boundary between two clusters and thus can be assigned to any of them. Such situations occur quite often while processing real data. The possible reason for this can be noise or lack of discriminatory power of the feature space, in which the patterns are represented [10].

Understanding of the described problem was a driving force for theoretical research in the field of clustering and eventually has led to the formalization of two main branches of cluster analysis: hard clustering and fuzzy clustering (see Figure 1.6).

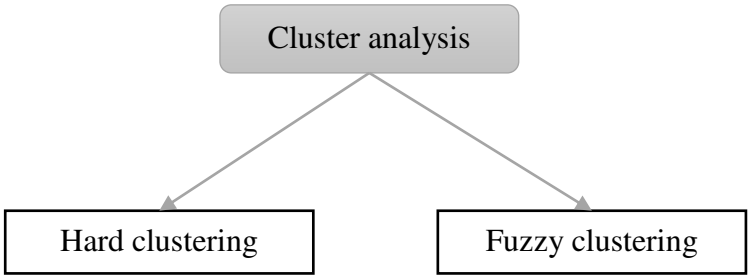


Figure 1.6. Classification of approaches for cluster analysis

Hard clustering (also crisp clustering) is a type of cluster analysis, that requires belonging of an object to one and only one cluster. It represents the classic approach that was described at the beginning of this section.

Mathematically, the result of hard clustering can be represented by a k × n matrix, that is called a hard k-partition of the data set D [4]:

$$U = \begin{pmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ u_{21} & u_{22} & \cdots & u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ u_{k2} & u_{k2} & \cdots & u_{kn} \end{pmatrix}, \tag{1.21}$$

where n denotes the number of patterns in the data set D, k denotes the number of clusters, and u_{ji} satisfies

$$u_{ji} = \{0, 1\}, \;\; 1 \leq j \leq k, \;\; 1 \leq i \leq n,$$

$$\sum_{j=1}^{k} u_{ji} = 1, \;\; 1 \leq i \leq n,$$

$$\sum_{i=1}^{n} u_{ji} > 0, \;\; 1 \leq j \leq k.$$

Fuzzy clustering (or soft clustering) is a type of cluster analysis, that implies belonging of an object to several clusters simultaneously, with different degrees of membership between 0 and 1 indicating their partial memberships [15].

The result of fuzzy clustering can be also represented by a $k \times n$ matrix, that is called a fuzzy k-partition of the data set D [4]:

$$M = \begin{pmatrix} \mu_{11} & \mu_{12} & \cdots & \mu_{1n} \\ \mu_{21} & \mu_{22} & \cdots & \mu_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \mu_{k2} & \mu_{k2} & \cdots & \mu_{kn} \end{pmatrix}, \tag{1.22}$$

where n denotes the number of patterns in the data set D, k denotes the number of clusters, and μ_{ji} denotes the degree of membership and satisfies

$$\mu_{ji} = [0,1], \;\; 1 \leq j \leq k, \;\; 1 \leq i \leq n,$$
$$\sum_{j=1}^{k} \mu_{ji} = 1, \;\; 1 \leq i \leq n,$$
$$\sum_{i=1}^{n} \mu_{ji} > 0, \;\; 1 \leq j \leq k.$$

Formulas (1.21)-(1.22) tell us only how to interpret the result of clustering, but we also need a mathematical description of clustering as a process. Clustering of a given data set D can be represented by an assignment function $f: D \to [0,1]^k, \;\; \mathbf{x} \to f(\mathbf{x})$, defined as follows:

$$f(x) = \begin{pmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \\ \vdots \\ f_k(\mathbf{x}) \end{pmatrix}, \tag{1.23}$$

where $f_i(\mathbf{x}) \in [0,1]$ for $i = 1, 2, \ldots, k$ and $\mathbf{x} \in D$, and

$$\sum_{i=1}^{k} f_i(x) = 1 \;\; \forall \mathbf{x} \in D.$$

If for every $\mathbf{x} \in D$, $f_i(\mathbf{x}) \in \{0,1\}$, then the clustering that is represented by f is a hard clustering; otherwise, it is a fuzzy clustering [4].

It should be noted that in the context of medical data even small deviations in the values of variables can be very important. Therefore, the threshold of sensitivity to the values deviations in the medical data analysis system should be as low as possible. For this reason, fuzzy clustering is preferable than hard clustering, due to the presence of membership degree of a pattern. Further in this diploma thesis we will focuses on fuzzy clustering.

CHAPTER 2. CLUSTERING ALGORITHMS

2.1. Fundamentals of clustering algorithms

In the previous chapter the goal of clustering was discussed in the context of partitioning a data set. Achieving of this goal is realized through a clustering algorithm – a process that usually consists of four sequential phases: data representation, constructing a criterion function, partitioning and assessment of output (Figure 2.1).

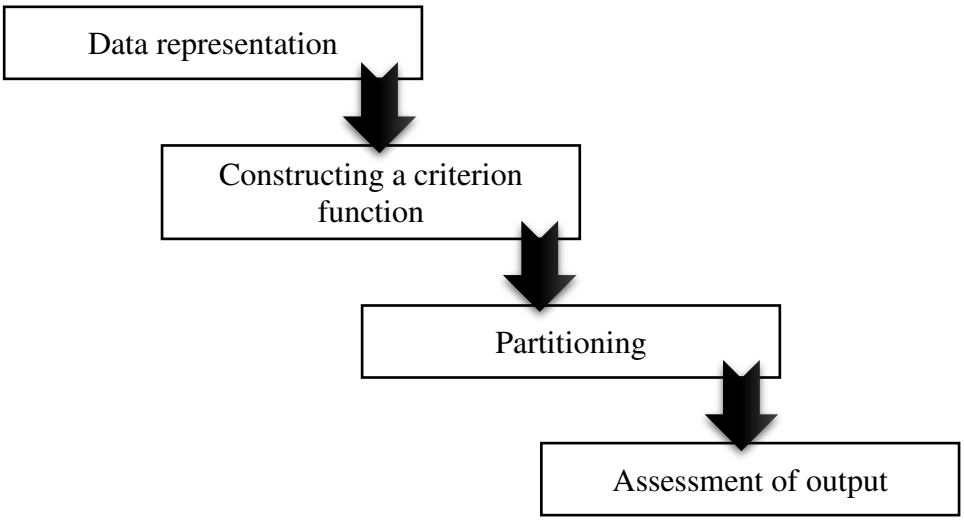
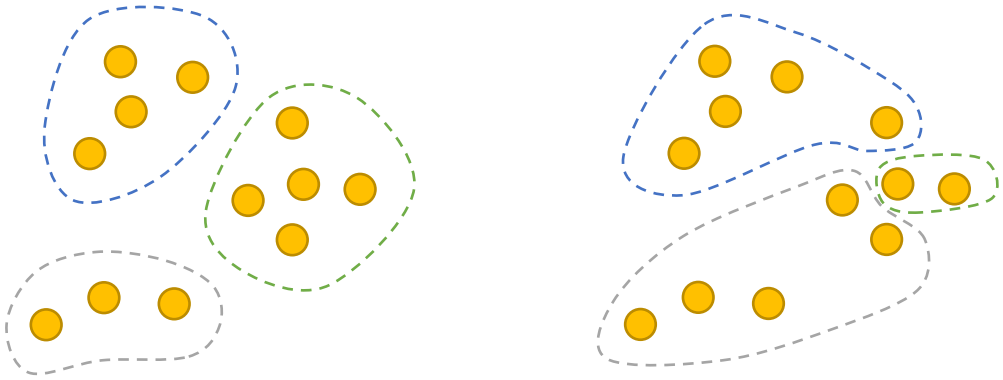


Figure 2.1. Phases of a clustering algorithm

During the ***data representation*** phase, information about the data set is clarified: number of initial patterns, data type, scale of data, and, optionally, number of classes or structure of clusters. This phase can involve either or both of feature selection and feature extraction to obtain an appropriate set of features. *The feature selection* is the process of identifying the most effective subset of the original features to use in clustering. *The feature extraction* is the use of one or more transformations of the input features to produce new salient features [5].

The ***constructing a criterion function*** phase is based on determining an appropriate similarity or dissimilarity measure for each attribute. The similarity (dissimilarity) measure

must be connected to the clustering algorithm either explicitly or implicitly. For the determined similarity measure clustering process can be understood as an optimization problem with a specific criterion function [6]. The criterion function determines a clustering quality (see Figure 2.2).



Good clustering **Bad clustering**

Figure 2.2. Criterion function impact on a clustering quality

During the ***partitioning*** phase the initial data set is partitioned into clusters. Each clustering algorithm defines its own way for performing partitioning. This way depends on a criterion function, constructed in the previous phase. In this phase the most of calculation, related to a clustering algorithm, is performed.

The ***assessment of output*** phase is the most important in a clustering algorithm, because in this phase an assessment of achieving a clustering goal and clustering accuracy is performed. The accuracy of clustering is usually assessed through the calculation of optimality criterions, or simply by comparing obtained clusters with classes of the data set, if they are known.

In the literature of cluster analysis, a plenty of optimality criterions, that are also called cluster validity indices, can be found. Some of them are listed in the Table 2.1.

Table 2.1. Optimality criterions of fuzzy clustering (Cluster Validity Indices)

Name	Formula	Description
Partition coefficient index (I_{PC}) [16]	$I_{PC} = \frac{1}{n}\sum_{j=1}^{k}\sum_{i=1}^{n}\mu_{ji}^{m}$ where μ_{ji} – the membership degree, see (1.22); m – the fuzzifier; n – the number of patterns in the data set; k – the number of clusters.	The index value is in the range $\left[\frac{1}{k}, 1\right]$. The optimal number of clusters k^* is calculated as $I_{PC}(k^*) = \max_{2\le k\le n-1} I_{PC}(k)$.
Partition entropy index (I_{PE}) [17]	$I_{PE} = -\frac{1}{n}\sum_{j=1}^{k}\sum_{i=1}^{n}\mu_{ji}\log_a(\mu_{ji})$ where a – the base of the logarithm; μ_{ji} – the membership degree, see (1.22); n – the number of patterns in the data set; k – the number of clusters.	The index value is in the range $[0, \log_a k]$. The optimal number of clusters k^* is calculated as $I_{PE}(k^*) = \min_{2\le k\le n-1} I_{PE}(k)$.
Fukuyama-Sugeno index (I_{FS}) [18]	$I_{FS} = \sum_{j=1}^{k}\sum_{i=1}^{n}\mu_{ji}^{m}\left(\left\|\mathbf{x}_i - \mathbf{z}_j\right\|^2 - \left\|\mathbf{z}_j - \mathbf{z}\right\|^2\right)$ where μ_{ji} – the membership degree, see (1.22); m – the fuzzifier; $\mathbf{x}_i$ – the i-th pattern from the initial data set; $\mathbf{z}$ – the mean of the initial data set; $\mathbf{z}_j$ – the mean of the j-th cluster; n – the number of patterns in the data set; k – the number of clusters.	The optimal number of clusters k^* is calculated as $I_{FS}(k^*) = \min_{2\le k\le n-1} I_{FS}(k)$
Xie-Beni index (I_{XB}) [5, 19]	$I_{XB} = \frac{\sum_{j=1}^{k}\sum_{i=1}^{n}\mu_{ji}^{m}\left\|\mathbf{x}_i - \mathbf{z}_j\right\|^2}{n \cdot \min_{q\ne j}\left\|\mathbf{z}_q - \mathbf{z}_j\right\|^2}$ where μ_{ji} – the membership degree, see (1.22); m – the fuzzifier; $\mathbf{x}_i$ – the i-th pattern from the initial data set; $\mathbf{z}_j$ – the mean of the j-th cluster; n – the number of patterns in the data set; k – the number of clusters.	The optimal number of clusters k^* is calculated as $I_{XB}(k^*) = \min_{2\le k\le n-1} I_{XB}(k)$

(continued)

Name	Formula	Description
PBMF index (I_{PBMF}) [20, 21]	$I_{PBMF} = \left(\frac{1}{k} \times \frac{\tilde{E}}{E_k} \times D_k\right)^2$ where $E_k = \sum_{j=1}^{k} \sum_{i=1}^{n} \mu_{ji} d(\mathbf{x}_i, \mathbf{z}_j)$; $\tilde{E} = \sum_{i=1}^{n} d(\mathbf{x}_i, \mathbf{z})$; $D_k = \max_{1 \le i < j \le k} d(\mathbf{z}_i, \mathbf{z}_j)$; μ_{ji} – the membership degree, see (1.22); $\mathbf{x}_i$ – the i-th pattern from the initial data set; $\mathbf{z}$ – the mean of the initial data set; $\mathbf{z}_j$ – the mean of the j-th cluster; n – the number of patterns in the data set; k – the number of clusters.	The optimal number of clusters k^* is calculated as $I_{PBMF}(k^*) = \max_{2 \le k \le n-1} I_{PBMF}(k)$.
Fuzzy hyper volume index (I_{FHV}) [5]	$I_{FHV} = \sum_{j=1}^{k} \sqrt{det(F_j)}$ where $F_j = \frac{\sum_{i=1}^{n} \mu_{ji}^m (\mathbf{x}_i - \mathbf{z}_j)(\mathbf{x}_i - \mathbf{z}_j)^T}{\sum_{i=1}^{n} \mu_{ji}^m}$ is the fuzzy covariance matrix of the j-th cluster; μ_{ji} – the membership degree, see (1.22); $\mathbf{x}_i$ – the i-th pattern from the initial data set; $\mathbf{z}_j$ – the mean of the j-th cluster; n – the number of patterns in the data set; k – the number of clusters.	The optimal number of clusters k^* is calculated as $I_{FHV}(k^*) = \min_{2 \le k \le n-1} I_{FHV}(k)$

Most of clustering algorithms define different approach to assessment of their output, which can be based on the calculation of some cluster validity index as well as a combination of several indices.

In the next sections we will make an overview of some existing clustering algorithms. Section 2.2 is dedicated to classical fuzzy clustering algorithms, the history of which was started with publication of a Fuzzy c-Means Clustering algorithm by Bezdek in 1981 (see Section 2.2.1). This algorithm has a variety of improvements and modifications for different fields of study, but we will consider only the two most well-known of them: a Gustafson-Kessel algorithm (Section 2.2.2) and a Gath-Geva clustering algorithm (Section 2.2.3). The significant limitation of all classical fuzzy clustering algorithms from the Fuzzy c-Means family is that they all assume the number of clustering to be previously known. In the Section 2.2.4 we propose a way of elimination of this limitation.

In Section 2.3 a Multi-Interval Discretization algorithm is described. It was originally proposed for performing a hard clustering, but a way to extend it to fuzzy clustering and author of this thesis describes it at the end of the Section.

A Fuzzy Entropy Based Fuzzy Classifier algorithm, described in Section 2.4, is promising, but it has some significant drawbacks, that should be eliminated. Author of this work proposes a way of eliminating mentioned drawbacks which are mentioned in Section 2.5.

2.2. Classical fuzzy clustering algorithms

2.2.1. Fuzzy c-Means Clustering

The Fuzzy c-Means (FCM) clustering algorithm was proposed by Bezdek [22] and can be considered as a generalization of ISODATA [23]. In this algorithm the number of clusters k, into which the initial data set is partitioned, assumed to be known. The goal of the FCM is to find an optimal fuzzy k-partition of the data set (1.22). For achieving this goal, the cost function is defined:

$$J_{FCM} = \sum_{j=1}^{k} \sum_{i=1}^{n} \mu_{ji}^{m} \left\| \mathbf{x}_i - \mathbf{z}_j \right\|^2 \tag{2.1}$$

where μ_{ji} – the membership degree; $m \in [1, \infty)$ – the fuzzifier, that is usually set to 2 (lager values of the fuzzifier favors fuzzier clusters); $\mathbf{x}_i$ – the i-th pattern from the initial data set; $\mathbf{z}_j$ – the mean of the j-th cluster; n – the number of patterns in the data set; k – the number of clusters.

The means of clusters and the membership degrees are iteratively updated until the cost function reaches its local minimum. Then the fuzzy k-partition of the data set is considered optimal.

The FCM algorithm determines following steps [6]*:*

Step 1. Set appropriate values for k, $m > 1$, $\varepsilon > 0$ (a threshold) and variable $t := 0$. Randomly initialize the cluster mean matrix $Z = [\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_k]$.

Step 2. Recalculate the membership degrees (elements of the fuzzy k-partition matrix M) for $j = 1,2,\dots,k$ and $i = 1,2,\dots,n$:

$$\mu_{ji}^{(t+1)} = \begin{cases} 1 \Big/ \left(\sum_{q=1}^{k} \left(\frac{\|\mathbf{x}_i - \mathbf{z}_j\|}{\|\mathbf{x}_i - \mathbf{z}_q\|} \right)^{2/(m-1)} \right), & if\ E_i = \emptyset \\ 1/|E_i|, & if\ E_i \neq \emptyset, j \in E_i \\ 0, & if\ E_i \neq \emptyset, j \notin E_i \end{cases}$$

where $E_i = \{j | j \in [1,k], \mathbf{x}_i = \mathbf{z}_j\}$.

Step 3. Update the cluster mean matrix Z for $j = 1,2,\dots,k$:

$$\mathbf{z}_j^{(t+1)} = \left(\sum_{i=1}^{n} \left(\mu_{ji}^{(t+1)} \right)^m \mathbf{x}_i \right) \Big/ \left(\sum_{i=1}^{n} \left(\mu_{ji}^{(t+1)} \right)^m \right)$$

Step 4. If $\|Z^{(t+1)} - Z^{(t)}\| \geq \varepsilon$ then put $t := t + 1$ and go to **Step 2.** Otherwise, the fuzzy k-partition $M^{(t)}$ is optimal.

The FCM algorithm has several significant drawbacks: convergence to an optimal solution is not ensured because the cluster mean matrix is initialized randomly [12]; sensitivity to noise and outliers that are forced into a cluster and used to calculate the cluster mean matrix [6]; the algorithm does not suppose a looking for the optimal number of clusters, moreover, the number of clusters must be previously known.

2.2.2. Gustafson-Kessel Clustering Algorithm

The Gustafson-Kessel (GK) clustering algorithm was proposed as an extension to the standard fuzzy c-means algorithm [24]. The algorithm is based on an adaptive distance measure, which makes possible a detection of clusters. It represented by different geometrical shapes. Detection of such clusters in one data set is not possible in the standard FCM, as it has predefined fixed topological structure and searches for clusters of only that shape in the data set. To provide an adaptive distance measure, in the GK algorithm the following inner-product norm is defined as [5]:

$$DGK_{ji}^2 = \left(\mathbf{x}_i - \mathbf{z}_j\right)^T A_j \left(\mathbf{x}_i - \mathbf{z}_j\right), \;\; 1 \le j \le k, \;\; 1 \le i \le n, \tag{2.2}$$

where $\mathbf{x}_i$ – the i-th pattern from the initial data set; $\mathbf{z}_j$ – the mean of the j-th cluster; n – the number of patterns in the data set; k – the number of clusters; A_j – the norm-inducing matrix of the j-th cluster, calculated as

$$A_j = \left[\rho_j det\left(F_j\right)\right]^{1/n} F_j^{-1}, \tag{2.3}$$

where ρ_j – the volume of the j-th cluster; F_j – the fuzzy covariance matrix of the j-th cluster, defined as:

$$F_j = \frac{\sum_{i=1}^{n} \mu_{ji}^m \left(\mathbf{x}_i - \mathbf{z}_j\right)\left(\mathbf{x}_i - \mathbf{z}_j\right)^T}{\sum_{i=1}^{n} \mu_{ji}^m} \tag{2.4}$$

where μ_{ji} – the membership degree; $m \in [1, \infty)$ – the fuzzifier (also called the weighting exponent), that is usually set to 2.

The cost function for the GK algorithm is similar as for the FCM except replacing a simple distance measure with the inner-product norm:

$$J_{GK} = \sum_{j=1}^{k} \sum_{i=1}^{n} \mu_{ji}^m DGK_{ji}^2 \tag{2.5}$$

The GK algorithm determines following steps [5]*:*

Step 1. Set appropriate values for the number of clusters $2 \le k \le n-1$, the weighting exponent $m > 1$, the threshold $\varepsilon > 0$ and variable $t := 0$. Randomly initialize the fuzzy k-partition (also called the partition matrix) $M^{(t)} = [\mu_{ji}^{(t)}]$ for $j = 1,2,\ldots,k$ and $i = 1,2,\ldots,n$ satisfying the conditions for (1.22).

Step 2. Calculate the cluster mean matrix $Z^{(t+1)} = \left[\mathbf{z}_1^{(t+1)}, \mathbf{z}_2^{(t+1)}, \ldots, \mathbf{z}_k^{(t+1)}\right]$:

$$\mathbf{z}_j^{(t+1)} = \left(\sum_{i=1}^{n} \left(\mu_{ji}^{(t)}\right)^m \mathbf{x}_i\right) \Big/ \left(\sum_{i=1}^{n} \left(\mu_{ji}^{(t)}\right)^m\right), \qquad 1 \le j \le k.$$

Step 3. Compute the cluster covariance matrices $F_j^{(t+1)}$ for $j = 1,2,\ldots,k$:

$$F_j^{(t+1)} = \frac{\sum_{i=1}^{n}\left(\mu_{ji}^{(t)}\right)^m \left(\mathbf{x}_i - \mathbf{z}_j^{(t+1)}\right)\left(\mathbf{x}_i - \mathbf{z}_j^{(t+1)}\right)^T}{\sum_{i=1}^{n}\left(\mu_{ji}^{(t)}\right)^m}$$

Step 4. Compute the inner-product norms for $j = 1,2,\ldots,k$ and $i = 1,2,\ldots,n$:

$$\left(DGK_{ji}^{(t+1)}\right)^2 = \left(\mathbf{x}_i - \mathbf{z}_j^{(t+1)}\right)^T \left[\rho_j det\left(F_j^{(t+1)}\right)\right]^{1/n} \left(F_j^{(t+1)}\right)^{-1}\left(\mathbf{x}_i - \mathbf{z}_j^{(t+1)}\right).$$

Step 5. Calculate the updated partition matrix $M^{(t+1)}$:

$$\mu_{ji}^{(t+1)} = 1\Big/\left(\sum_{q=1}^{k}\left(\frac{DGK_{ji}^{(t+1)}}{DGK_{qi}^{(t+1)}}\right)^{2/(m-1)}\right), \quad 1 \le j \le k, \quad 1 \le i \le n.$$

Step 6. If $\left\|M^{(t+1)} - M^{(t)}\right\| \ge \varepsilon$ then put $t := t+1$, $M^{(t)} := M^{(t+1)}$ and go to **Step 2.** Otherwise, the partition matrix $M^{(t)}$ is optimal.

In general, the GK algorithm has the same drawbacks as the FCM, except the sensitivity to noise and outliers, which is less in the GK thanks to the adaptive distance measure.

2.2.3. Gath-Geva Clustering Algorithm

The Gath-Geva (GG) clustering algorithm, also known as the fuzzy maximum likelihood estimation (FMLE) algorithm, was proposed as an extension to the Gustafson-Kessel clustering algorithm [25]. Like the GK algorithm, the GG algorithm is based on an adaptive distance measure, but, instead of the inner-product norm, the distance norm, based on the fuzzy maximum likelihood estimates, is used [5]:

$$DGG_{ji}^2 = \frac{(2\pi)^{\left(\frac{n}{2}\right)}\sqrt{det\left(F_j\right)}}{\alpha_j} \cdot e^{\left(\frac{1}{2}\left(\mathbf{x}_i - \mathbf{z}_j\right)^T F_j^{-1}\left(\mathbf{x}_i - \mathbf{z}_j\right)\right)}, \quad 1 \le j \le k, \quad 1 \le i \le n \qquad (2.6)$$

where $\mathbf{x}_i$ – the i-th pattern from the initial data set; $\mathbf{z}_j$ – the mean of the j-th cluster; n – the number of patterns in the data set; k – the number of clusters; F_j – the fuzzy covariance matrix of the j-th cluster; α_j – the prior probability of selecting the j-th cluster, defined as:

$$\alpha_j = \frac{1}{n}\sum_{i=1}^{n} \mu_{ji} \tag{2.7}$$

where μ_{ji} – the membership degree.

The Gath–Geva clustering algorithm determines following steps [5]*:*

Step 1. Set appropriate values for the number of clusters $2 \le k \le n-1$, the weighting exponent $m > 1$, the threshold $\varepsilon > 0$ and variable $t := 0$. Randomly initialize the fuzzy k-partition (also called the partition matrix) $M^{(t)} = [\mu_{ji}^{(t)}]$ for $j = 1,2,\ldots,k$ and $i = 1,2,\ldots,n$ satisfying the conditions for (1.22).

Step 2. Calculate the cluster mean matrix $Z^{(t+1)} = \left[\mathbf{z}_1^{(t+1)}, \mathbf{z}_2^{(t+1)}, \ldots, \mathbf{z}_k^{(t+1)}\right]$:

$$\mathbf{z}_j^{(t+1)} = \left(\sum_{i=1}^{n} \left(\mu_{ji}^{(t)}\right)^m \mathbf{x}_i\right) \Big/ \left(\sum_{i=1}^{n} \left(\mu_{ji}^{(t)}\right)^m\right), \qquad 1 \le j \le k.$$

Step 3. Compute the cluster covariance matrices $F_j^{(t+1)}$ for $j = 1,2,\ldots,k$:

$$F_j^{(t+1)} = \frac{\sum_{i=1}^{n} \left(\mu_{ji}^{(t)}\right)^m \left(\mathbf{x}_i - \mathbf{z}_j^{(t+1)}\right)\left(\mathbf{x}_i - \mathbf{z}_j^{(t+1)}\right)^T}{\sum_{i=1}^{n} \left(\mu_{ji}^{(t)}\right)^m}$$

Step 4. Compute the distance norm for $j = 1,2,\ldots,k$ and $i = 1,2,\ldots,n$:

$$\left(DGG_{ji}^{(t+1)}\right)^2 = \frac{(2\pi)^{\left(\frac{n}{2}\right)}\sqrt{det\left(F_j^{(t+1)}\right)}}{\alpha_j^{(t)}} \cdot e^{\left(\frac{1}{2}\left(\mathbf{x}_i - \mathbf{z}_j^{(t+1)}\right)^T \left(F_j^{(t+1)}\right)^{-1} \left(\mathbf{x}_i - \mathbf{z}_j^{(t+1)}\right)\right)}$$

where $\alpha_j^{(t)} = \frac{1}{n}\sum_{i=1}^{n} \mu_{ji}^{(t)}$.

Step 5. Calculate the updated partition matrix $M^{(t+1)}$:

$$\mu_{ji}^{(t+1)} = 1 \Big/ \left(\sum_{q=1}^{k} \left(\frac{DGG_{ji}^{(t+1)}}{DGG_{qi}^{(t+1)}}\right)^{2/(m-1)}\right), \quad 1 \le j \le k, \quad 1 \le i \le n.$$

Step 6. If $\left\|M^{(t+1)} - M^{(t)}\right\| \ge \varepsilon$ then put $t := t+1$, $M^{(t)} := M^{(t+1)}$ and go to **Step 2.** Otherwise, the partition matrix $M^{(t)}$ is optimal.

The GG algorithm can detect clusters of varying shapes, sizes and densities. This is because the cluster volumes are not predefined, and the cluster covariance matrix is used in conjunction with a distance norm, that involves an exponential term. The main drawback of the clustering algorithm is following: it converges to a near local optimum because of the exponential distance norm, thus the result of clustering depends on the initialization method [5].

2.2.4. Determining the optimal number of clusters

As were mentioned above, the significant limitation of all classical fuzzy clustering algorithms from the Fuzzy c-Means family is that they all assume the number of clustering to be previously known. But in most cases, we do not know this information when we get a data set. Therefore, the good idea is to determine the optimal number of clusters during the clustering process by selecting one among the set of possible numbers of clusters. Such functionality can be provided by employing some cluster validity index to a clustering algorithm.

Several well-known CVIs are listed above in the Table 2.1. In addition to them, very promising is the Pairing Frequency cluster validity index (I_{PF}), proposed by Hongyan Cui, Kuo Zhang et al. [26]. The index is based on the analysis of a pairing frequency – the phenomenon, when a pair of patterns in the initial datasets always belongs to the same cluster regardless of the number of clusters. Let us describe the procedure of calculating I_{PF}, proposed in [26], in more details. At first, we need to obtain a fuzzy k-partition of the initial data set of n patterns (see Formula 1.22):

$$M = [\mu_{ji}], \quad 1 \leq j \leq k, 1 \leq i \leq n \tag{2.8}$$

The value of j when μ_{ji} reaches its maximum for the i-th pattern is assigned as p_i. For any pair of patterns $\mathbf{x}_s$, $\mathbf{x}_t$ ($1 \leq s, t \leq n$), if $p_s = p_t = c$, then both $\mathbf{x}_s$ and $\mathbf{x}_t$ belongs to the same cluster c.

At second, we define a pattern matrix F_k for k clusters:

$$F_k = \begin{bmatrix} f_{11} & f_{12} & \cdots & f_{1n} \\ f_{21} & f_{22} & \cdots & f_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ f_{n1} & f_{n2} & \cdots & f_{nn} \end{bmatrix} \tag{2.9}$$

An element f_{st} of the pattern matrix F_k indicates a degree of belonging of $\mathbf{x}_s$ and $\mathbf{x}_t$ patterns to the same cluster or different clusters:

$$f_{st} = \begin{cases} 1 - \dfrac{k}{k-1} \times |\max \boldsymbol{\mu}_s - \max \boldsymbol{\mu}_t| & if \;\; p_s = p_t \\ -\dfrac{k}{2k-2} \times \left|\max \boldsymbol{\mu}_s + \max \boldsymbol{\mu}_t - \dfrac{2}{k}\right| & if \;\; p_s \neq p_t \end{cases} \tag{2.10}$$

where $\max \boldsymbol{\mu}_s = \max\{\mu_{js} : 1 \leq j \leq k\}$; $\max \boldsymbol{\mu}_t = \max\{\mu_{jt} : 1 \leq j \leq k\}$.

In case $0 < f_{st} \leq 1$, most of membership of $\mathbf{x}_s$ and $\mathbf{x}_t$ should be placed in the same cluster. The closer the value of f_{st} to 1, the greater the degree of belonging of the patterns to the same cluster. In case $-1 \leq f_{st} < 0$, least of membership of $\mathbf{x}_s$ and $\mathbf{x}_t$ should be placed in the same cluster. The closer the value of f_{st} to -1, the greater the degree of belonging of the patterns to different clusters.

Next, we calculate the final global pairwise pattern matrix Q:

$$Q = \frac{\sum_{k=2}^{k_{upper}} F_k}{k_{upper} - 1} \tag{2.11}$$

where $k_{upper} = \max\left(k_{max}, \lfloor 0.5\sqrt{n} \rfloor\right)$, k_{max} – the maximal number of clusters into which the initial data set can be partitioned.

The matrix Q indicates a degree of belonging of any two patterns to the same cluster or different clusters regardless of the number of clusters.

Finally, the pairing frequency index I_{PF}, proposed in [26], is defined as:

$$I_{PF} = S(Q \circ F_k) \tag{2.12}$$

where '$\circ$' represents the Hadamard product and S represents the sum of all elements of matrix $Q \circ F_k$.

The optimal number of clusters k^* is calculated as

$$I_{PF}(k^*) = \max_{2 \le k \le k_{max}} I_{PF}(k). \tag{2.13}$$

The considered cluster validity index, based on the pairing frequency, in comparison with the most of well-known indices gives more accurate results [26], because of avoiding of using compactness-to-separation ratio criteria and using information from several clustering processes (2.11) to compute the value of I_{PF}.

2.3. Multi-Interval Discretization

The Multi-Interval Discretization (MID) algorithm, proposed by Denis V. Popel [27], is based on an information density concept. The author defines a discretization as "a process of transforming values of a continuous variable into a finite number of intervals and associating with each interval a discrete numerical value". Obviously, the discretization in this context is the same as the clustering.

The MID algorithm divides the data set into clusters, depending on values of a specific attribute of each pattern. A continuous variable, that represents this specific attribute, is denoted by s. It can take values from the set $S = \{s_1, s_2, \dots, s_n\}$ in the range $T = \left[T^{min}, T^{max}\right]$, where n – the number of patterns represented in the set S, $T^{min} < T^{max}$ and $\Delta T = T^{max} - T^{min}$. A partition $\mathcal{P}$ of the set S into k intervals (clusters) can be defined as [27]:

$$\begin{cases} \mathcal{P} = \bigcup_{j=1}^{k} \mathcal{P}_j \\ \mathcal{P}_j \neq \emptyset \quad for\ j = 1,2,\dots,k \\ s_i \notin \mathcal{P}_g\ if\ s_i \in \mathcal{P}_j \quad for\ j \neq g \end{cases} \tag{2.14}$$

where $\mathcal{P}_j = \left[\mathcal{P}_j^{min}, \mathcal{P}_j^{max}\right]$ is the partition of the j-th cluster, that can be described by taking T^{min} and T^{max} into consideration:

$$\begin{cases} \mathcal{P}_1^{min} = T^{min} \\ \mathcal{P}_j^{min} > T^{min} & j = 2,3,\dots,k-1 \\ \mathcal{P}_j^{max} < T^{max} & j = 2,3,\dots,k-1 \\ \mathcal{P}_{j+1}^{min} = \mathcal{P}_j^{max} & j = 1,2,\dots,k-1 \\ \mathcal{P}_k^{max} = T^{max} \end{cases} \tag{2.15}$$

The information measures defined in [27] are represented by the entropy $H(T,S)$ and the information density $D(T,S)$ of the partition $\mathcal{P}$:

$$H(T,S) = -\sum_{i=0}^{n} \delta_i \cdot \log \delta_i \,, \tag{2.16}$$

$$D(T,S) = \begin{cases} H(T,S)/\log(n+1), & if\ n > 0 \\ 0, & if\ n = 0 \end{cases} \tag{2.17}$$

where δ_i the probabilities, calculated as

$$\delta_i = \begin{cases} \left(s_1 - T^{min}\right)/\Delta T \,, & if\ i = 0, \\ (s_{i+1} - s_i)/\Delta T \,, & if\ i = 1,2,\dots,n-1, \\ (T^{max} - s_n)/\Delta T \,, & if\ i = n. \end{cases} \tag{2.18}$$

Next, the MID algorithm employs the conditional information density $D(T,S|T^{cut})$ of the partition $\mathcal{P}$, given by the cut point T^{cut} that splits the partition $\mathcal{P}$ into two partitions $\mathcal{P}_1$ and $\mathcal{P}_2$:

$$D(T,S|T^{cut}) = p_1 \cdot D(T_1,S_1) + p_2 \cdot D(T_2,S_2), \tag{2.19}$$

where $D(T_1,S_1)$ – the information density of the partition $\mathcal{P}_1$; $D(T_2,S_2)$ – the information density of the partition $\mathcal{P}_2$; $p_1 = \left(T^{cut} - T^{min}\right)/\Delta T$; $p_2 = (T^{max} - T^{cut})/\Delta T$.

The cut points are determined according to the information density optimization criterion:

$$T^{cut} = \text{argmax}\{D(T,S|T^{cut}) - D(T,S)\} \tag{2.20}$$

The MID clustering algorithm starts with the single initial partition $\mathcal{P}$ and splits it recursively. The result of clustering is a set of clusters, boundaries of which are unambiguously defined by a set of accepted cut points.

The Multi-Interval Discretization algorithm determines following steps [27]*:*

Step 1. Sort all values from the set $S = \{s_1, s_2, \ldots, s_n\}$ in ascending order. Define boundaries T^{min}, T^{max} of the initial partition $\mathcal{P}$.

Step 2. Generate a set of possible cut points with a kernel $\Delta T/100$ for the current partition $\mathcal{P}$. Calculate the information density $D(T,S)$ according to formula (2.17).

Step 3. Form combinations of partitions $\mathcal{P}_1$ and $\mathcal{P}_2$ for all possible cut points and calculate the resulting conditional information density $D(T,S|T^{cut})$ according to (2.19).

Step 4. Find a potential optimal cut point T^{cut} according to the information density optimization criterion, defined in formula (2.20).

Step 5. If the resulting conditional information density $D(T,S|T^{cut})$, calculated for potential optimal cut point T^{cut}, is greater than the information density $D(T,S)$ of the current partition $\mathcal{P}$, then accept the cut point T^{cut} and continue with **Step 6**. Otherwise, terminate the recursion and go to **Step 7**.

Step 6. Execute Steps 2-5 for both $\mathcal{P}_1$ and $\mathcal{P}_2$ recursively.

Step 7. Sort all cut points, which are boundaries between clusters. Let the number of cut points be denoted by m, then $k^* = m + 1$ is the optimal number of clusters.

In his paper [27] Popel illustrates the above algorithm with the following example. The initial data set to be clustered is

$$S = \{0.022, 0.376, 0.443, 0.519, 0.598, 0.704, 0.837, 0.841, 0.899, 0.953, 0.954\}.$$

All values from the set are in the range $T = [0, 1]$. In Figure 2.3 two steps of clustering are shown. On the left part of the Figure 2.3 the distributions of the information density gain ($D(T,S|T^{cut}) - D(T,S)$) for different cut points are depicted. The right part illustrates the final cut points $T^{cut} = \{0.080, 0.807\}$ according to the optimization criterion, defined in (2.20) [27].

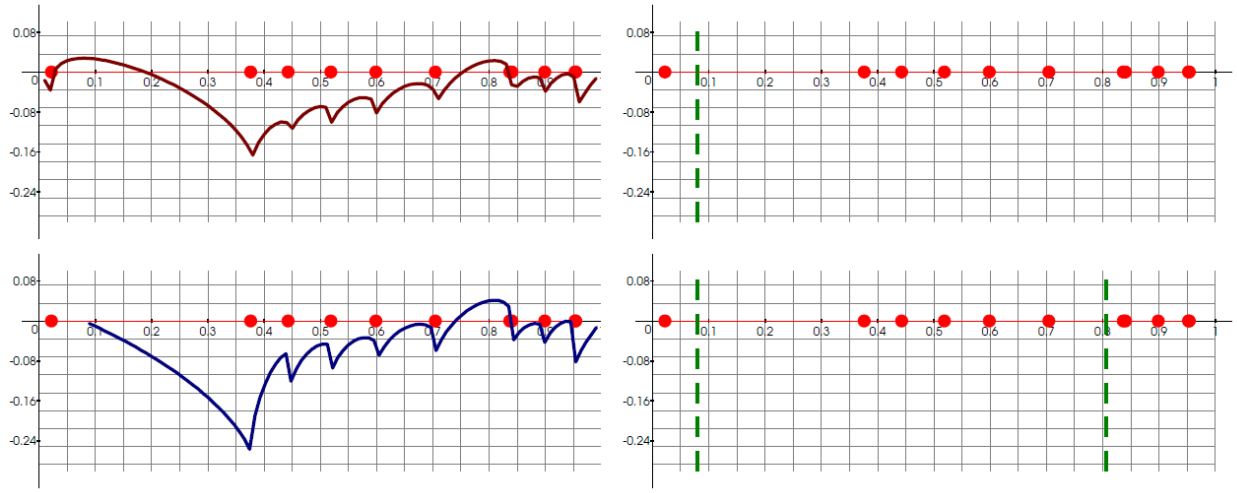


Figure 2.3. Example of the MID clustering algorithm, given by Popel [27]

The main drawback of the algorithm is that a calculation of the kernel $\Delta T/100$ does not involve a scale of data, that can lead to inaccurate results. For example, if attribute values are in the range $[1, 500]$ our cut point step (the kernel) is $(500 - 1)/100 = 4.99$, and if the optimal cut point equals to $T^{cut} = 13$, we will not be able to discover it, because the nearest considered possible cut points are 10.98 and 15.97.

Originally the Multi-Interval Discretization algorithm was proposed as a hard-clustering algorithm, but it can be extended to provide fuzzy clustering through several additional computations, described below.

The MID algorithm returns the optimal set of clusters, defined by their boundaries. Let this set of boundaries be denoted by $B = \{b_1, b_2, \ldots, b_k, b_{k+1}\}$, where k is the number of clusters; $b_1 = T^{min}$; $b_{k+1} = T^{max}$; $b_j \in T^{cut}$ for $j = 2,3, \ldots k$. Next, to be able to compute the fuzzy k-partition, we need to compute a set of cluster centers $C = \{c_1, c_2, \ldots, c_w\}$ (see Figure 2.4), where $k \leq w \leq 2k - 2$ is the number of cluster centers.

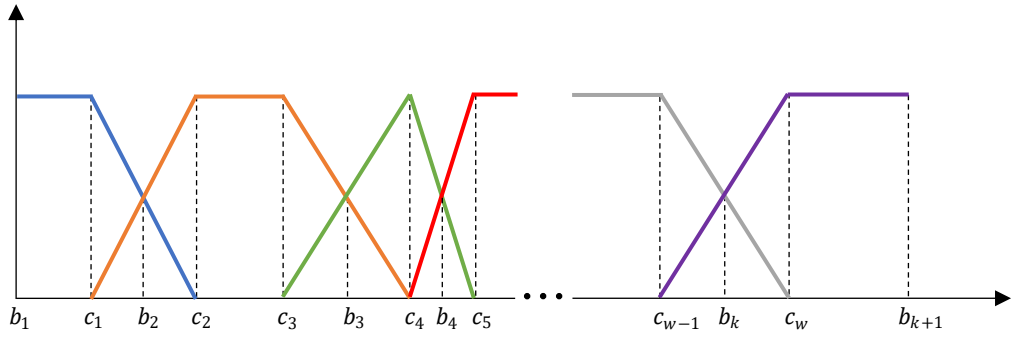


Figure 2.4. The fuzzy membership function with boundaries and centers of clusters

Cluster centers are such values, the defines a space, where patterns fully belong to some cluster. Many ways for defining cluster centers can be found, but in this diploma thesis we propose a one simple method, that allows to compute a set of cluster centers with a maximal possible cardinality $|C| = 2k - 2$. This approach assumes, that the set can contain duplicates $c_{2l} = c_{2l+1}$ for $1 \leq l \leq k - 2$, that is possible if some cluster have only one cluster center (for example, a cluster represented by the green chart in Figure 2.4, i.e. the c_4 value is present in the C twice). A cluster center for $1 \leq j \leq k - 1$ is defined by:

$$c_{2j-1} = b_{j+1} - \Delta r_j, \qquad c_{2j} = b_{j+1} + \Delta r_j \tag{2.21}$$

where $\Delta r_j = \min\left(\frac{b_{j+1}-b_j}{2}, \frac{b_{j+2}-b_{j+1}}{2}\right)$.

The membership function (shown in Figure 2.4) for converting a continuous value of the variable s to fuzzy values is defined for each of k clusters by [28]:

- for the first (left-most) cluster and for the k-th (right-most) cluster

$$\mu_1 = \begin{cases} 1, & \text{for } x \leq c_1 \\ \dfrac{c_2 - x}{c_2 - c_1}, & \text{for } c_1 < x \leq c_2 \\ 0, & \text{otherwise} \end{cases} \tag{2.22}$$

$$\mu_k = \begin{cases} 0, & \text{for } x < c_{2k-3} \\ \dfrac{x - c_{2k-3}}{c_{2k-2} - c_{2k-3}}, & \text{for } c_{2k-3} \leq x < c_{2k-2} \\ 1, & \text{otherwise} \end{cases} \tag{2.23}$$

- for the q-th cluster, where $q = 2,3,\dots,k-1$

$$\mu_q = \begin{cases} 0, & \text{for } x \le c_{2q-3} \\ \dfrac{x - c_{2q-3}}{c_{2q-2} - c_{2q-3}}, & \text{for } c_{2q-3} < x \le c_{2q-2} \\ 1, & \text{for } c_{2q-2} < x \le c_{2q-1} \\ \dfrac{c_{2q} - x}{c_{2q} - c_{2q-1}}, & \text{for } c_{2q-1} < x \le c_{2q} \\ 0, & \text{otherwise} \end{cases} \tag{2.24}$$

Using the membership function, defined above (2.22) – (2.24), for the initial S we are able to calculate the fuzzy k-partition $M = [\mu_{ji}]$, $1 \le j \le k$, $1 \le i \le n$. Thus, with the described computation we can force the MID algorithm to make fuzzy clustering.

2.4. Fuzzy Entropy Based Fuzzy Classifier

The Fuzzy Entropy Based Fuzzy Classifier (FEBFC) algorithm with Feature Selection was proposed by Hahn-Ming Lee, Chih-Ming Chen et al. [29]. This algorithm uses fuzzy entropy, based on Shannon's entropy, as a criterion of optimality. According to the proposed approach, the fuzzy entropy $FE(\tilde{A})$ is defined on the universal set $X = \{r_1, r_2, \dots, r_n\}$, where $i = 1, 2, \dots, n$, for the elements within an interval (cluster) in a non-probabilistic way:

$$FE(\tilde{A}) = \sum_{j=1}^{m} FE_{C_j}(\tilde{A}) = \sum_{j=1}^{m} -D_j \log_2 D_j \tag{2.25}$$

where $\tilde{A}$ is a fuzzy set defined on an interval of pattern space which contains k elements ($k < n$); $C_1, C_2, \dots, C_m$ represent m classes into which the n elements are divided; $FE_{C_j}(\tilde{A})$ is the fuzzy entropy of the elements of class j in an interval, defined as $FE_{C_j}(\tilde{A}) = -D_j \log_2 D_j$; D_j is the match degree with fuzzy set $\tilde{A}$ for the elements of class j in an interval,

where $j = 1, 2, \dots, m$, defined as $D_j = \frac{\sum_{r \in S_{C_j}(r_n)} \mu_{\tilde{A}}(r)}{\sum_{r \in X} \mu_{\tilde{A}}(r)}$; $\mu_{\tilde{A}}(r_i)$ is the mapped membership degree of the element r_i with the fuzzy set $\tilde{A}$; $S_{C_j}(r_n)$ is a set of elements of class j on the universal set X (subset of the universal set X).

The fuzzy entropy cluster validity index (I_{FE}), also known as the total fuzzy entropy, is defined as a sum of fuzzy entropies of all clusters:

$$I_{FE} = \sum_{i=1}^{k} FE_i^* \tag{2.26}$$

where k – the number of clusters; FE_i^* – the fuzzy entropy of the i-th cluster, calculated as a sum of fuzzy entropies of all fuzzy sets on the i-th interval. Then the optimal number of clusters k^* is calculated as

$$I_{FE}(k^*) = \min_{2 \le k \le n-1} I_{FE}(k) \tag{2.27}$$

The proposed fuzzy entropy measure [29] satisfies following four De Luca-Termini axioms, generalized by Kosko [30]. Therefore, it can be considered as well-defined:

1) $E(A) = 0$ if and only if $A \in 2^X$ (A is a nonfuzzy set; E is an entropy measure);
2) $E(A) = 1$ if and only if $m_A(x_i) = 0.5$ for all i ($m_A(x_i)$ is a membership degree of x_i);
3) $E(A) \le E(B)$ if A is less fuzzy then B, i.e., if $m_A(x) \le m_B(x)$ when $m_B(x) \le 0.5$ and $m_A(x) \ge m_B(x)$ when $m_B(x) \ge 0.5$;
4) $E(A) = E(A^c)$.

The FEBFC algorithm includes following steps [29]*:*

Step 1. Set the initial numbers of intervals (clusters) $k \coloneqq 2$.

Step 2. Locate the centers of intervals:

2A. Set initial cluster centers $c_1, c_2, \dots c_k$. (each cluster has only one cluster center in the FEBFC). They can be randomly selected from x_i, $i = 1,2, \dots n$, or $c_q = \frac{q-1}{k-1}$, $q = 1,2, \dots k$.

2B. Assign a cluster label to each element (the smallest Euclidean distance):

$$\left|x_i - c_q^*\right| = \min_{1 \le q \le k} \left|x_i - c_q\right|$$

2C. Recompute the cluster centers.

$$c_q = \frac{\sum_{i=1}^{n_q} x_i^q}{n_q}$$

where n_q is the total number of patterns x_i^q, that belong to q-th cluster.

2D. Compare recomputed cluster centers with previous. If any center was changed then go to **Step 2B.** Otherwise, go to **Step 3.**

Step 3. Assign the membership function for each interval according to:

$$\mu_1 = \begin{cases} 1, & \text{for } x \le c_1 \\ \frac{c_2 - x}{c_2 - c_1}, & \text{for } c_1 < x \le c_2 \\ 0, & \text{otherwise} \end{cases} \qquad \mu_q = \begin{cases} 0, & \text{for } x \le c_{q-1} \\ \frac{x - c_{q-1}}{c_q - c_{q-1}}, & \text{for } c_{q-1} < x \le c_q \\ \frac{c_{q+1} - x}{c_{q+1} - c_q}, & \text{for } c_q < x \le c_{q+1} \\ 0, & \text{otherwise} \end{cases}$$

$$\mu_k = \begin{cases} 0, & \text{for } x < c_{k-1} \\ \frac{x - c_{k-1}}{c_k - c_{k-1}}, & \text{for } c_{k-1} \le x < c_k \\ 1, & \text{otherwise} \end{cases}$$

where $q = 2,3,\ldots,k-1$.

Step 4. Compute the index I_{FE} for k clusters and $k-1$ clusters.

Step 5. If $I_{FE}(k) < I_{FE}(k-1)$, then partition again ($k := k+1$) and go to **Step 2**; otherwise, $k-1$ is the optimal number of clusters.

For illustrating the FEBFC algorithm, we use the following example. Let X be a distribution of three classes of objects represented by values of some attribute of these objects. The distribution divided into three and four intervals is shown in Figure 2.5 (a) and (b) respectively as a set of objects △, □ and ○, placed on x axis. Position on the axis corresponds with a value of the attribute.

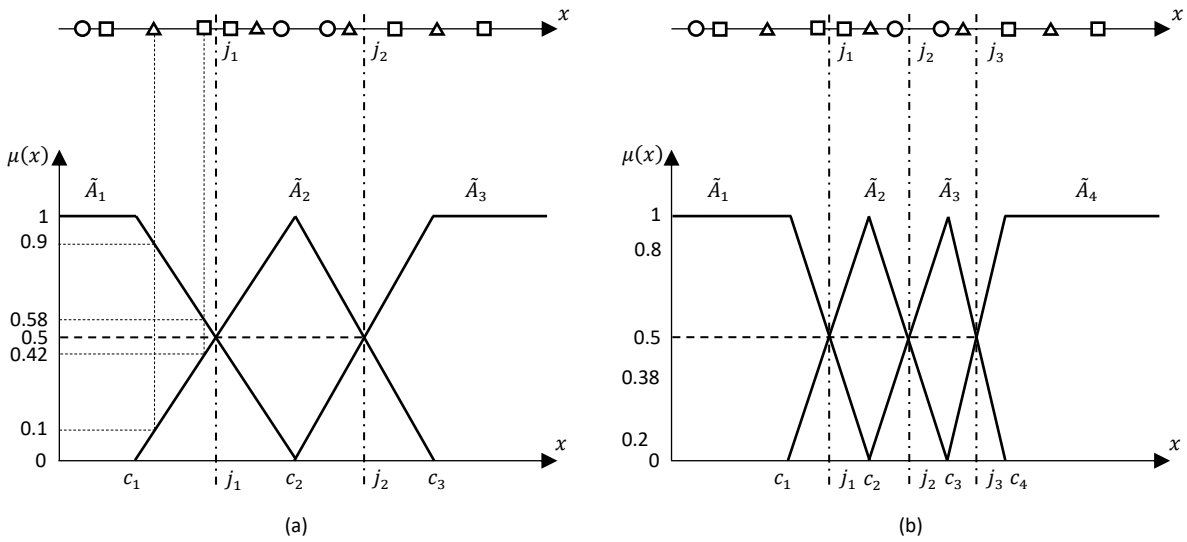


Figure 2.5. Example of a distribution of 3 classes of objects (△, □ and ○ denote class 1, class 2 and class 3 respectively) with corresponding membership functions

In the first considered case (Figure 2.5 (a)), the distribution is divided into three intervals, that are $(-\infty; j_1)$, $[j_1; j_2)$ and $[j_2; \infty)$. On these intervals fuzzy sets $\tilde{A}_1$, $\tilde{A}_2$ and $\tilde{A}_3$ are obtained using a membership function. The centers of fuzzy sets are denoted as c_1, c_2, c_3. Let us introduce a calculation process of the total fuzzy entropy of the distribution with mentioned dividing.

In the below paragraphs the sequence of steps needed to calculate the fuzzy entropy measure of the interval $(-\infty; j_1)$ is detailly described. The measure calculation process for other intervals is skipped, because the same method is used.

At first, we find a total membership degree for each class on values of the fuzzy set $\tilde{A}_1$ from the interval:

- total membership degree of "△" is 0.9;
- total membership degree of "□" is $1 + 0.58 = 1.58$;
- total membership degree of "○" is 1.

Then we calculate match degrees:

- $D_{\triangle} = 0.9/(0.9 + 1.58 + 1) = 0.9/3.48 = 0.25862$;
- $D_{\square} = 1.58/3.48 = 0.45402$;
- $D_{\circ} = 1/3.48 = 0.28736$.

In the next step we calculate fuzzy entropies of $\tilde{A}_1$ on $(-\infty; j_1)$:

- $FE_{\triangle}(\tilde{A}_1) = -0.25862 \times \log_2 0.25862 = 0.50459$;
- $FE_{\square}(\tilde{A}_1) = -0.45402 \times \log_2 0.45402 = 0.51721$;
- $FE_{\circ}(\tilde{A}_1) = -0.28736 \times \log_2 0.28736 = 0.51698$;
- $FE_1(\tilde{A}_1) = FE_{\triangle}(\tilde{A}_1) + FE_{\square}(\tilde{A}_1) + FE_{\circ}(\tilde{A}_1) = 1.53878$.

Similarly, the fuzzy entropies of $\tilde{A}_2$ and $\tilde{A}_3$ on $(-\infty; j_1)$ are calculated:

- $FE_1(\tilde{A}_2) = FE_{\triangle}(\tilde{A}_2) + FE_{\square}(\tilde{A}_2) + FE_{\circ}(\tilde{A}_2) = 0.70627$;
- $FE_1(\tilde{A}_3) = FE_{\triangle}(\tilde{A}_3) + FE_{\square}(\tilde{A}_3) + FE_{\circ}(\tilde{A}_3) = 0$.

The fuzzy entropy of the interval $(-\infty; j_1)$ equals:

$$FE_1^* = FE_1(\tilde{A}_1) + FE_1(\tilde{A}_2) + FE_1(\tilde{A}_3) = 2.24505.$$

Similarly, the fuzzy entropies FE_2^* and FE_3^* of the corresponding intervals $[j_1; j_2)$ and $[j_2; \infty)$ can be obtained:

$$FE_2^* = FE_2(\tilde{A}_1) + FE_2(\tilde{A}_2) + FE_2(\tilde{A}_3) = 3.67795;$$

$$FE_3^* = FE_3(\tilde{A}_1) + FE_3(\tilde{A}_2) + FE_3(\tilde{A}_3) = 0.95096.$$

Finally, the total fuzzy entropy of the distribution (a) is calculated in the following way:

$$I_{FE}^{(a)} = FE_1^* + FE_2^* + FE_3^* = 2.24505 + 3.67795 + 0.95096 = 6.87396.$$

In the second considered case (Figure 2.5 (b)), the distribution is divided into four intervals, that are $(-\infty; j_1)$, $[j_1; j_2)$, $[j_2; j_3)$ and $[j_3; \infty)$. Using the method, gradually described before, we calculate the total fuzzy entropy measure for this case of dividing of the distribution:

$$I_{FE}^{(b)} = 1.56636 + 2.30609 + 1.43211 + 0.94244 = 6.247.$$

According to obtained results ($I_{FE}^{(b)} < I_{FE}^{(a)}$), dividing the distribution into four intervals is preferable, that is obvious from the Figure 2.5.

The proposed clustering algorithm [29] has significant drawback: the cluster validity index is calculated as a simple summation of the fuzzy entropy measures of all intervals, into which the set is divided. This solution would be perfect if all intervals had equal length and quantity of patterns on them. But usually there are several clusters of different size among one set. Distances between the patterns are also different. Thus, a simple addition of fuzzy entropy values of intervals may lead to inaccurate results.

2.5. Fuzzy Information Density Based Fuzzy Classifier

For eliminating the drawback, described in Section 2.4., the author of this thesis proposes to use the information density measure, mentioned in Section 2.3, for estimating of classification instead of the fuzzy entropy measure. This approach brings to consideration distances between different patterns. Replacing the entropy measure by the fuzzy entropy measure in formula (2.17), we obtain the following definition of the fuzzy information density FD_q of the q-th cluster ($q = 1,2,\ldots,k$):

$$FD_q = \begin{cases} FE_q / \log_2(n_q + 1), & \text{if } n_q > 0 \\ 0, & \text{if } n_q = 0 \end{cases} \tag{2.28}$$

where FE_q – the fuzzy entropy on the q-th interval; n_q – the number of patterns on the q-th interval.

The fuzzy entropy cluster validity index (I_{FE}) should be then replaced by the fuzzy information density cluster validity index (I_{FD}), that is also called the total information density, defined as

$$I_{FD} = \sum_{q=1}^{k} \omega_q \times FD_q \tag{2.29}$$

where FD_q – the fuzzy information density on the q-th interval; $\omega_q = \frac{n_q}{n}$ is a weight coefficient; n – the number of patterns in the data set; n_q – the number of patterns on the q-th interval. Then the optimal number of clusters k^* is calculated as

$$I_{FD}(k^*) = \min_{2 \le k \le n-1} I_{FD}(k) \tag{2.30}$$

A new clustering algorithm was obtained as a result of applying the proposed changes. Author of the thesis has called it a Fuzzy Information Density Based Fuzzy Classifier (FIDBFC).

The FIDBFC algorithm (modification of [29]*) determines following steps:*

Step 1. Set the initial number of clusters (intervals) $k \coloneqq 2$.

Step 2. Locate the centers of intervals using following subsequence of steps:

2A. Find the initial centers of intervals $c_1, c_2, \dots, c_k$ using formula:

$$c_q = x_{min} + (x_{max} - x_{min}) \times \frac{q-1}{k-1}, \qquad q = 1,2, \dots, k.$$

2B. Assign each element of the distribution to a corresponding interval with the smallest Euclidian distance to the interval center:

$$\left|x_i - c_q^*\right| = \min_{1 \le q \le k} \left|x_i - c_q\right|$$

where c_q^* is the closest center to the element x_i.

2C. Recompute the cluster centers.

$$c_q = \frac{\sum_{i=1}^{n_q} x_i^q}{n_q}$$

where n_q is the total number of patterns x_i^q, that belong to q-th cluster.

2D. Compare recomputed cluster centers with previous. If any center was changed then go to **Step 2B.** Otherwise, go to **Step 3.**

Step 3. Assign the membership function for each interval according to:

$$\mu_1 = \begin{cases} 1, & \text{for } x \le c_1 \\ \frac{c_2 - x}{c_2 - c_1}, & \text{for } c_1 < x \le c_2 \\ 0, & \text{otherwise} \end{cases} \qquad \mu_q = \begin{cases} 0, & \text{for } x \le c_{q-1} \\ \frac{x - c_{q-1}}{c_q - c_{q-1}}, & \text{for } c_{q-1} < x \le c_q \\ \frac{c_{q+1} - x}{c_{q+1} - c_q}, & \text{for } c_q < x \le c_{q+1} \\ 0, & \text{otherwise} \end{cases}$$

$$\mu_k = \begin{cases} 0, & \text{for } x < c_{k-1} \\ \frac{x - c_{k-1}}{c_k - c_{k-1}}, & \text{for } c_{k-1} \le x < c_k \\ 1, & \text{otherwise} \end{cases}$$

where $q = 2,3, \dots, k-1$.

Step 4. Compute the $I_{FD}(k)$ for k clusters and $I_{FD}(k-1)$ for $k-1$ clusters according to formula (2.29).

Step 5. If $I_{FD}(k) < I_{FD}(k-1)$, then partition again ($k \coloneqq k+1$) and go to **Step 2**; otherwise, $k-1$ is the optimal number of clusters.

For illustrating this approach, the example from Section 2.4 can be used. After obtaining the fuzzy entropy value of the interval $(-\infty; j_1)$, we are able to calculate the information density measure:

$$FD_1 = FE_1^*/\log_2(k+1) = 2.24505/\log_2 5 = 0.96689.$$

The information density measures of the intervals $[j_1; j_2)$ and $[j_2; \infty)$ are calculated in a same way:

$$FD_2 = FE_2^*/\log_2(k+1) = 3.67795/\log_2 6 = 1.42283;$$
$$FD_3 = FE_3^*/\log_2(k+1) = 0.95096/\log_2 4 = 0.47548.$$

Finally, according to (2.29) the total fuzzy information density measure of the distribution (a) is calculated in the following way:

$$I_{FD}^{(a)} = (4/12) \times 0.96689 + (5/12) \times 1.42283 + (3/12) \times 0.47548 =$$
$$= 1.03401.$$

In the second considered case (Figure 2.5 (b)), the distribution is divided into four intervals, that are $(-\infty; j_1)$, $[j_1; j_2)$, $[j_2; j_3)$ and $[j_3; \infty)$. Using the same method, as described before, the total fuzzy information density measure for this case of dividing of the distribution:

$$I_{FD}^{(b)} = (3/12) \times 0.78318 + (4/12) \times 0.99318 + (2/12) \times 0.90356 +$$
$$+ (3/12) \times 0.47122 = 0.79526.$$

According to obtained results ($I_{FD}^{(b)} < I_{FD}^{(a)}$), dividing the distribution into four intervals is preferable, like according to the FEBFC algorithm. To approve the advantages of FIDBFC algorithm, a comparation of algorithms on medical data will be described in Chapter 4.

CHAPTER 3. A SOFTWARE TOOL FOR DATA ANALYSIS BASED ON CLUSTERING

3.1. Design of the Fuzzy Clustering Tool

Achieving the goal of this thesis involves development of a software tool that would be able to make transformation of values of any numeric attribute of the medical data set into fuzzy values. The main requirement for the tool is the ability of integration into an expert system for medical data analysis, in a way that it could be considered as a computational component of the system.

The functional requirements can be summarized in the following list:

- reading data set from a file;
- basic analysis of the initial data set;
- graphical visualization of basic analysis results;
- fuzzy clustering using algorithms from Chapter 2;
- importing clustering results from external software;
- graphical visualization of clustering results;
- fuzzification of the initial data set depending on clustering results;
- write fuzzification result to a file.

According to the above list, the most of functionality of the tool is related to fuzzy clustering. Therefore, it was decided that the name of the developed tool will be a Fuzzy Clustering Tool.

A Use Case diagram of the Fuzzy Clustering Tool, based on the above list of functional requirements, is shown in Figure 3.1. According to the Use Case diagram, a Class diagram for the Fuzzy Clustering Tool was designed, that can be found in Appendix 1. Each class will be described in detail in the following sections.

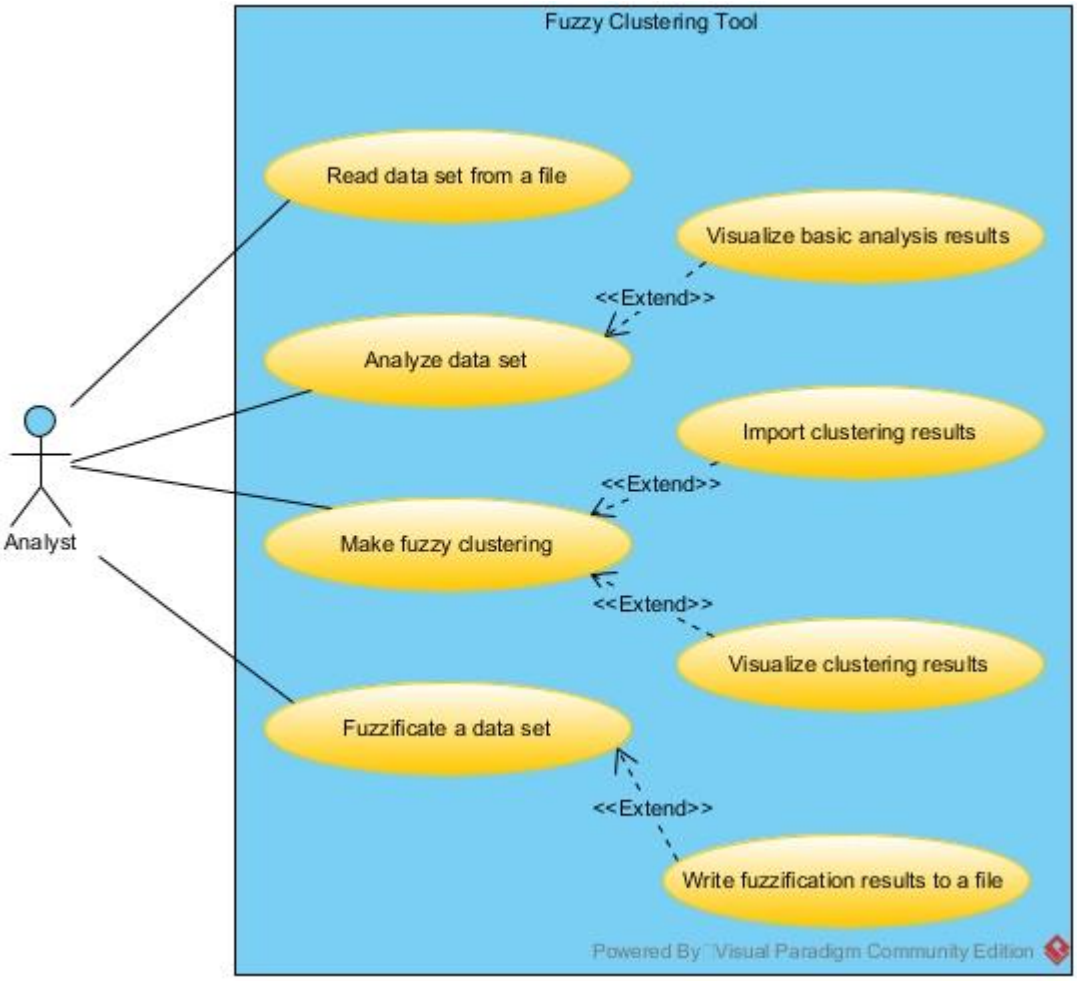


Figure 3.1. The Use Case diagram of the Fuzzy Clustering Tool.

It was decided to separate a GUI from core of application to the tool more flexible for changes. It gives a possibility to easily replace the GUI implementation by any other or use an application core as an external library in other applications. Thus, the Fuzzy Clustering Tool consists of two logical modules, implementation details of which described in Sections 3.2 – 3.3:

- core module: contains business logic of the application, all core functionality of the application; represents Model and Controller layers according to MVC pattern;
- GUI module: allows an analyst (a user) to use features of the tool; represents View layer according to MVC pattern.

3.2. Core module implementation

The core module was implemented in C++ programming language (ISO/IEC 14882:2017). Visual Studio 2017 was used as an IDE. Git was used as a version control system.

A structure of a **FuzzyClusteringTool** "solution" is shown in Figure 3.2. This "solution" consists of two projects, representing the logical modules, mentioned in Section 3.1.1. The **CoreFuzzyClustering** project here is an implementation of the core module.

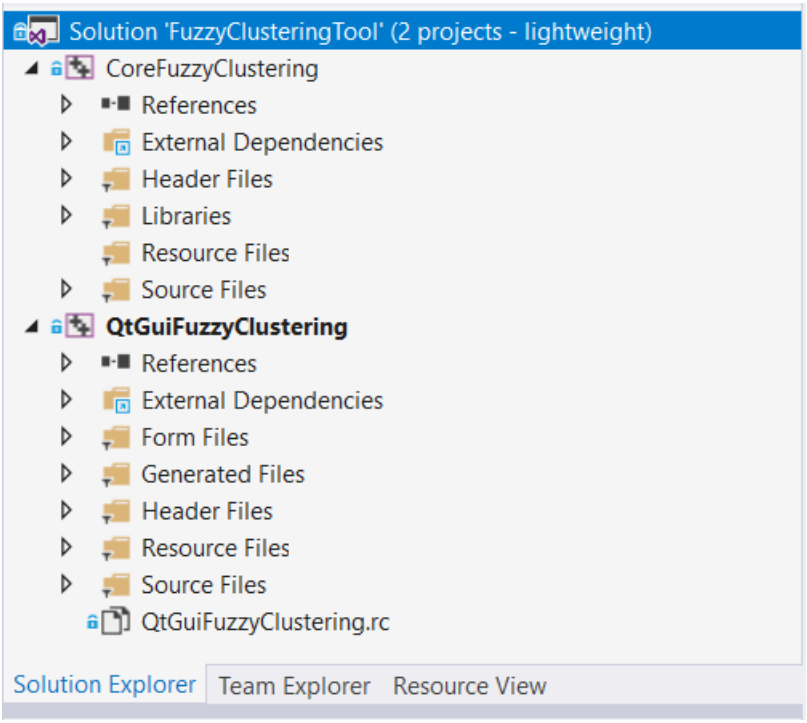


Figure 3.2. A structure of the Fuzzy Clustering Tool implementation as a "solution" in Visual Studio 2017.

Two external libraries were required for implementing the Fuzzy Clustering Tool: one for parsing *.xml* files and one for logging. A lot of C++ libraries for these purposes can be found on the Internet, so few requirements were specified to select appropriate solutions: free, open-source, lightweight, easy to use, cross-platform. Depending on mentioned criteria a selection was made and following libraries were chosen: **tinyxml2** for parsing *.xml* files

and **easylogging++** for logging in application. The source code of the mentioned libraries was included into the CoreFuzzyClustering project under the ***libraries*** directory.

Describing of the implementation details of the CoreFuzzyClustering project should be started from a **Solver** class (see Figure 3.3). This class provides an access to all functionality of the core module and plays a role of Controller according to MVC pattern. The Solver class has only one field ***_dataset***, which is a pointer to an object of type **Dataset** (will be described later). The object is created in a default constructor and deleted in a default destructor.

Figure 3.3. A Solver class view.

The Solver class contains following methods:

- ***readInitialDataset()*** – reads the data set, described in the *.xml* file, path to which is given as a parameter of the method;
- ***makeClustering()*** – runs a clustering algorithm, defined by the parameter;
- ***makeImportClustersDetails()*** – imports clustering results, obtained from an external tool, from *.xml* file with appropriate structure;
- ***makeFuzzification()*** – runs fuzzification using current values of borders and centers of clusters;
- ***writeFuzzyDataToFile()*** – writes fuzzy data set to specified file;
- ***isAttrNumeric()*** – checks if the attribute with given index is numeric;
- ***getAttrCount()*** – gets the total number of attributes;
- ***getInputAttrCount()*** – gets the number of input attributes;
- ***getOutputAttrCount()*** – gets the number of output attributes;
- ***getAttrId()*** – gets an id of the attribute with given index;
- ***getAttrMode()*** – gets a mode (input or output) of the attribute with given index;
- ***getAttrType()*** – gets a type (numeric or nominal) of the attribute with given index;
- ***getAttrTitle()*** – gets a title of the attribute with given index;
- ***getDatasetName()*** – gets a name of the data set;
- ***getInstancesCount()*** – gets the number of instances (patterns) in the dataset;
- ***getAttrDistinctCount()*** – gets the number of instances with distinct values of attribute;
- ***getAttrUniqueCount()*** – gets the number of instances with unique values of attribute;
- ***getAttrUniquePerc()*** – gets the percentage of instances with unique values of attribute;
- ***getAttrMissingCount()*** – gets the number of instances with missing values of attribute;
- ***getAttrMissingPerc()*** – gets the percentage of instances with missing values of attribute;
- ***getAttrClustersCount()*** – gets the number of clusters of the attribute;
- ***getAttrBordersInString()*** – gets a string containing all borders values;
- ***getNumericAttrCount()*** – gets the number of numeric attributes;
- ***getNumericAttrMinValue()*** – gets minimal value of the numeric attribute;

- ***getNumericAttrMaxValue()*** – gets maximal value of the numeric attribute;
- ***getNumericAttrAvgValue()*** – gets average value of the numeric attribute;
- ***getNumericAttrStdDevValue()*** – gets standard deviation for numeric attribute;
- ***getPossibleNominalAttrValuesList()*** – gets list of possible values of the nominal attribute;
- ***getNominalAttrCountOfValues()*** – gets the number of duplicating of specified value of the attribute in the data set;
- ***getNominalAttrPercOfValues()*** – gets the percentage of duplicating of specified value of the attribute in the data set;
- ***getAttrClustersCentersById()*** – gets list of cluster centers of the attribute.

The Solver class uses three enumerators, shown in Figure 3.4: **ClusteringAlgorithm** enumerates all implemented algorithms in the core module (these algorithms were described in Chapter 2); **Mode** enumerates possible modes of attributes (an attribute can be either *input* or *output*); **Type** enumerates possible types of attributes (an attribute can be either *numeric* or *nominal*).

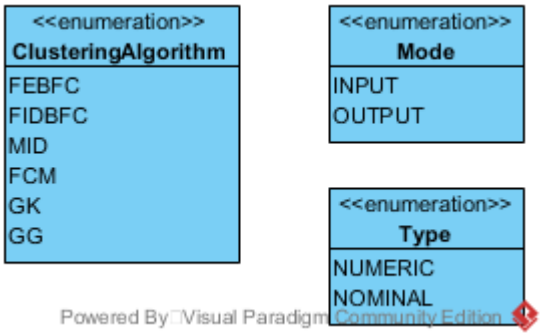


Figure 3.4. Enumerators in the core module.

The Solver class has an association relationship with a **Dataset** class, that is shown in Appendix 1. The Dataset class represents a data set, analyzed in the Fuzzy Clustering Tool. A diagram with the Dataset class description is shown in Figure 3.5. The following methods in the Dataset class have the same name as mentioned above methods from the Solver class have: *readInitialDataset()*, *isAttrNumeric()*, *getAttrMode()*, *getAttrType()*, *getNumericAttrCount()*, *getInputAttrCount()*, *getOutputAttrCount()*, *getDatasetName()*,

getAttrDistinctCount(), *writeFuzzyDataToFile()*, *getAttrTitle()*, *getAttrUniqueCount()*, *getAttrUniquePerc()*, *getAttrMissingCount()*, *getInstancesCount()*, *getAttrMissingPerc()*, *getNumericAttrMinValue()*, *getNumericAttrMaxValue()*, *getNumericAttrAvgValue()*, *getNumericAttrStdDevValue()*, *getAttrCount()*, *getPossibleNominalAttrValuesList()*, *getAttrClustersCount()*, *getAttrBordersInString()*, *getNominalAttrCountOfValues()*, *getNominalAttrPercOfValues()*. The reason is that these methods are just invoked from the Solver class (author of such design tried to separate interfaces from realizations), so their functionality is the same and will not be described here again to avoid duplicity.

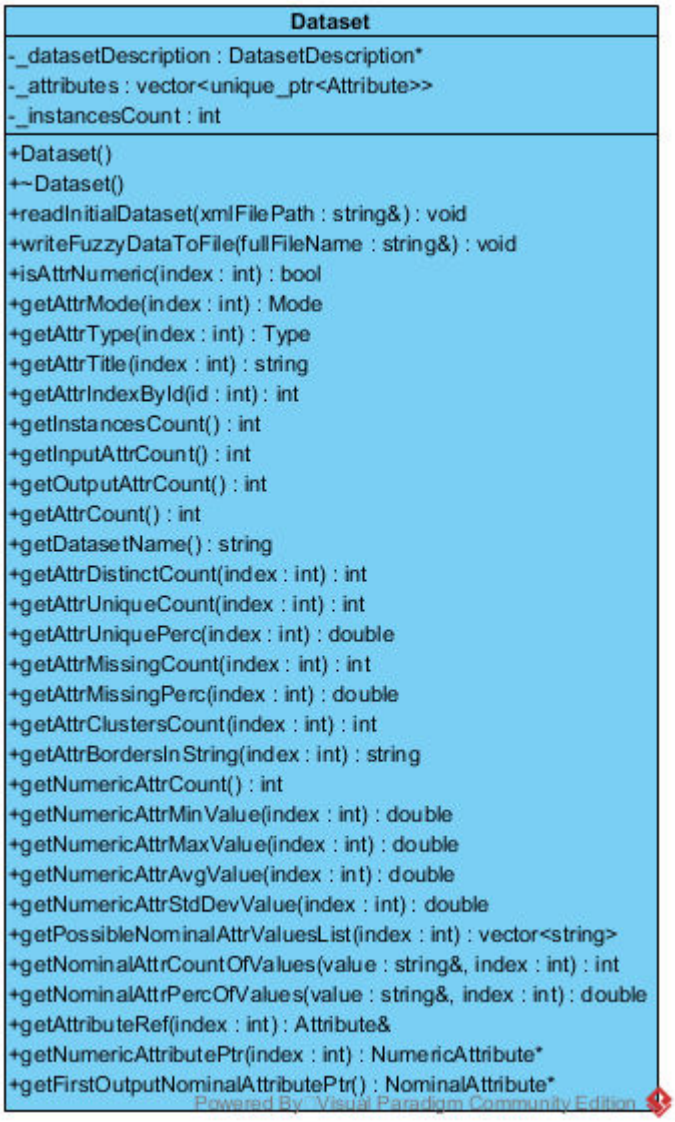


Figure 3.5. A Dataset class view.

The Dataset class contains following fields: ***_datasetDescription*** – a pointer to an object of type **DatasetDescription**, which contains information related to the data set; ***_attributes*** – a list of pointers to objects of **Attribute** type, which contain data of a particular attribute; ***_instancesCount*** – the number of instances (patterns) in the data set.

The Dataset class also contains following methods:

- ***getAttrIndexById()*** – gets index of the attribute by given id;
- ***getAttributeRef()*** – gets a reference on the attribute with given index;
- ***getNumericAttributePtr()*** – gets a pointer to the NumericAttribute object (NumericAttribute class derives from Attribute class) if the attribute with given index is numeric, or *nullptr* otherwise;
- ***getFirstOutputNominalAttributePtr()*** – gets a pointer to the first found in _attributes list a NominalAttribute object (NominalAttribute class derives from Attribute class), that is also output, or *nullptr* if no nominal output attribute is found.

A **DatasetDescription** class provides a description for the dataset, represented by the Dataset class (see Figure 3.6). It contains following fields: ***_dsName*** – a name of the data set; ***_dsSourcePath*** – full path to the data file, from which the data set is read; ***_inputAttrCount*** – the number of input attributes; ***_outputAttrCount*** – the number of output attributes; ***_totalAttrCount*** – the total number of attributes in the data set. Also DatasetDescription class has one public method ***readDatasetDescriptionXml()***, that reads the .xml file with data set description.

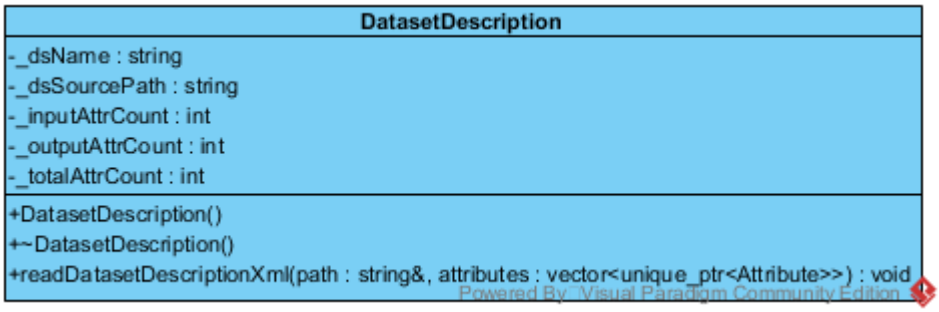


Figure 3.6. A DatasetDescription class view.

An **Attribute** class is an abstract class and represents an attribute of any type. It has two protected fields ***_attributeHeader*** and ***_attributeData***, which are pointed to corresponding objects with information, that describes an attribute. Also the Attribute class includes 2 constructors, 1 virtual destructor, 13 defined public methods and 1 pure virtual method to avoid instantiating of this class (see Figure 3.7).

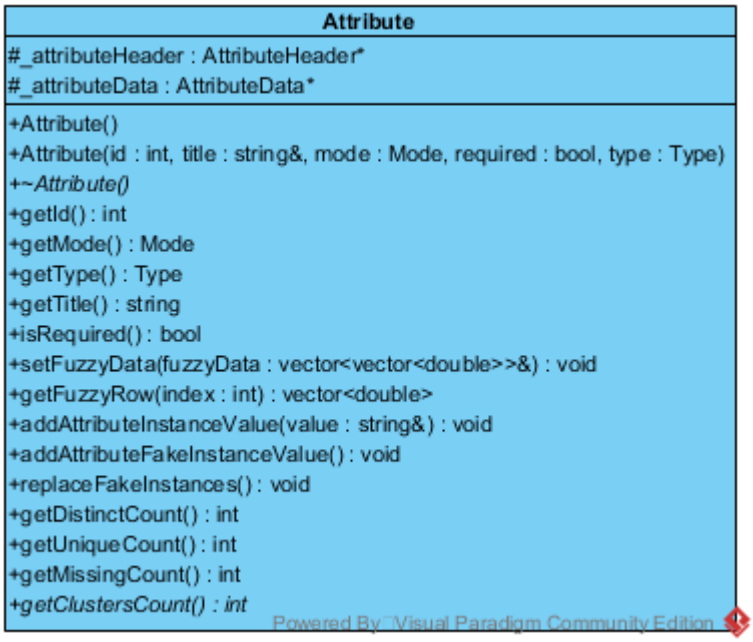


Figure 3.7. An Attribute class view.

The methods ***getId()***, ***getMode()***, ***getType()***, ***getTitlle()*** and ***getRequired()*** of the **Attribute** class are getters for corresponding fields in **AttributeHeader** class, that are ***_id***, ***_mode***, ***_type***, ***_title*** and ***_required***, as shown in Figure 3.8.

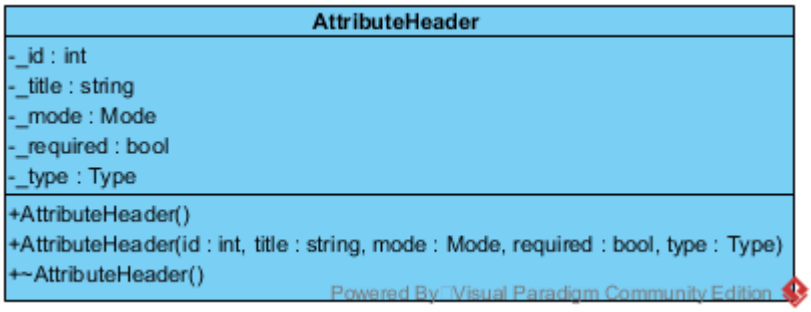


Figure 3.8. An AttributeHeader class view.

The Attribute class also has following methods:

- ***setFuzzyData()*** – sets fuzzy data of this attribute for all patterns;
- ***getFuzzyRow()*** – gets fuzzy values of this attribute for given pattern;
- ***addAttributeInstanceValue()*** – adds value of this attribute to the data set;
- ***addAttributeFakeInstanceValue()*** – adds fake value of this attribute to the data set (in case missing value in data file);
- ***replaceFakeInstances()*** – replaces fake values with some significant values (defined in AttributeData realizations);
- ***getDistinctCount()*** – gets the number of distinct values;
- ***getUniqueCount()*** – gets the number of unique values;
- ***getMissingCount()*** – gets the number of missing values;
- ***getClustersCount()*** – gets the number of clusters (pure virtual method).

The Attribute class is extended by a **NumericAttribute** class and a **NominalAttribute** class. The NumericAttribute class is shown in Figure 3.9. It has three own fields, that are ***_clusterCount*** (the number of clusters), ***_clusterCenterList*** (the list of centers of clusters) and ***_clusterBorderList*** (the list of borders of clusters).

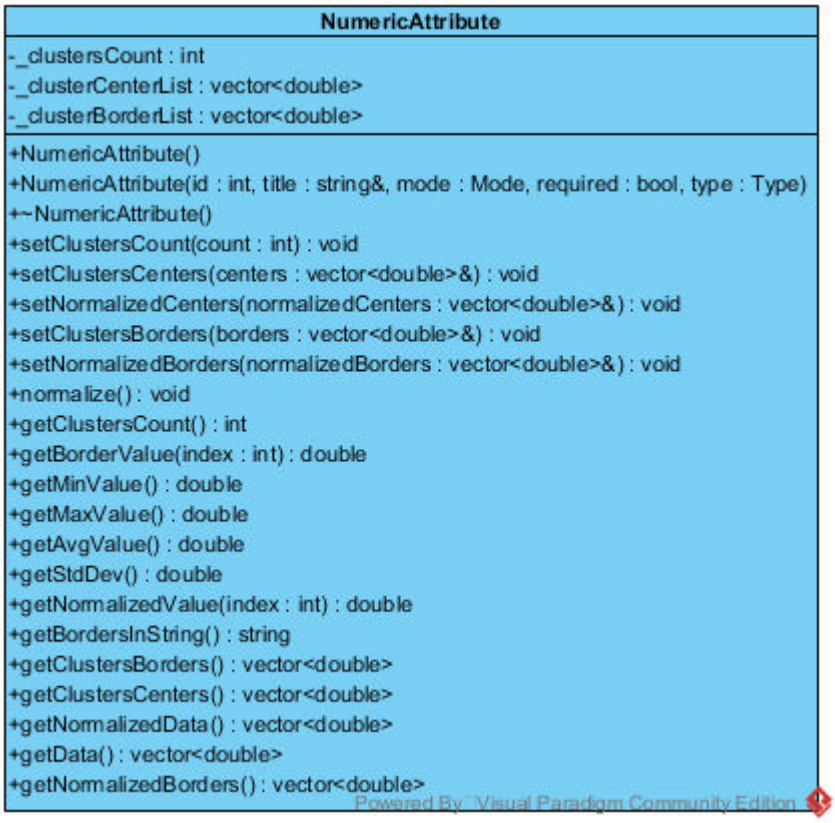


Figure 3.9. A NumericAttribute class view.

The NumericAttribute class also has following own methods:

- ***setClustersCount()*** – sets the number of clusters;
- ***setClustersCenters()*** – sets values of centers of clusters according to given list;
- ***setNormalizedCenters()*** – sets values of centers of clusters according to given list of normalized values;
- ***setClustersBorders()*** – sets values of borders of clusters according to given list;
- ***setNormalizedBorders()*** – sets values of borders of clusters according to given list of normalized values;
- ***normalize()*** – make normalization of attribute values;
- ***getBorderValue()*** – gets a border value by given index in list;
- ***getMinValue()*** – gets a minimal value of this attribute;
- ***getMaxValue()*** – gets a maximal value of this attribute;
- ***getAvgValue()*** – gets an average value of this attribute;
- ***getStdDev()*** – gets a standard deviation for this attribute;
- ***getValue()*** – gets an attribute value of the pattern with given index;

- ***getNormalizedValue()*** – gets a normalized attribute value of the pattern with given index;
- ***getBordersInString()*** – gets borders of this attribute in string;
- ***getClustersBorders()*** – gets list of borders of clusters;
- ***getClustersCenters()*** – gets list of centers of clusters;
- ***getNormalizedData()*** – gets list of normalized attribute values;
- ***getData()*** – gets list of attribute values;
- ***getNormalizedBorders()*** – gets list of normalized borders of clusters.

The **NominalAttribute** class is shown in Figure 3.10. It has only one own field ***_possibleValuesList*** (the list of possible values of a nominal attribute). The NominalAttribute class contains following own public methods:

- ***computePossibleValuesList()*** – fills *_possibleValuesList* container with possible values of this nominal attribute;
- ***getCountOfValueUsage()*** – gets the number of repeating of given value among all attribute values;
- ***getValue()*** – gets an attribute value of the pattern with given index;
- ***getPossibleValuesList()*** – gets the list of possible values of this attribute.

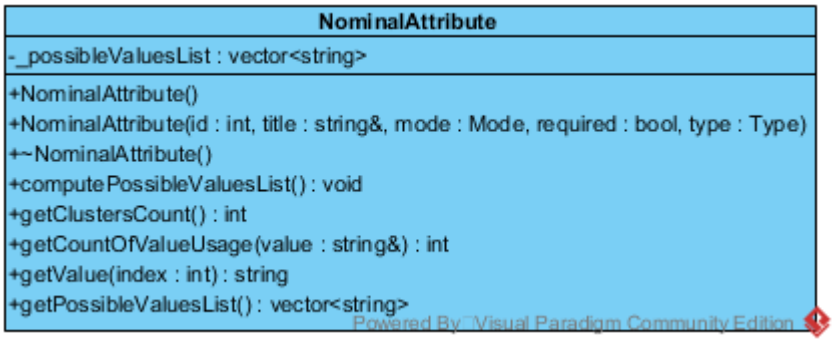


Figure 3.10. A NominalAttribute class view.

An **AttributeData** class aggregates all data (including data that were read from data file as well as fuzzy data) of an attribute. This class has three protected fields, that are ***_missingCount*** (the number of missing values of the attribute in the initial data set), ***_fakeValuesIndexes*** (indices of such missing values, that need to be faked for correct data

input) and ***_attributeFuzzyData*** (fuzzified data of the attribute). The AttributeData class contains following methods:

- ***getDistinctCount()*** – functionality delegated from the method with the same name in the Attribute class (pure virtual method);
- ***getUniqueCount()*** – functionality delegated from by the method with the same name in the Attribute class (pure virtual method);
- ***addAttributeInstanceValue()*** – functionality delegated from by the method with the same name in the Attribute class (pure virtual method);
- ***addAttributeFakeInstanceValue()*** – functionality delegated from by the method with the same name in the Attribute class (pure virtual method);
- ***replaceFakeInstances()*** – functionality delegated from by the method with the same name in the Attribute class (pure virtual method);
- ***getMissingCount()*** – functionality delegated from by the method with the same name in the Attribute class.

The AttributeData class is the base class for an **AttributeDataNumeric** and an **AttributeDataNominal** classes as shown in Figure 3.11. The AttributeDataNumeric class has seven own field, that are ***_attributeData*** (list of numeric values of the attribute), ***_normalizedData*** (list of normalized values of the attribute), ***_map*** (map container in which the key is an attribute value converted to integer with some accuracy and the value is number of repeating of this value in the set), ***_minValue*** (minimal value of the attribute), ***_maxValue*** (maximal value of the attribute), ***_statSum*** (sum of all non-missed values of the attribute) and ***_statCount*** (the number of all non-missed values of the attribute). The AttributeDataNumeric class defines ***getMinValue()***, ***getMaxValue()***, ***getAvgValue()***, ***getStdDev()*** and ***getValue()*** methods with delegated functionalities from the methods with the same names defined in the NumericAttribute class. In addition, the AttributeDataNumeric class defines ***normalizeData()*** method, which functionality is delegated from the *normalize()* method of the NumericAttribute class, and overrides virtual methods of the base class.

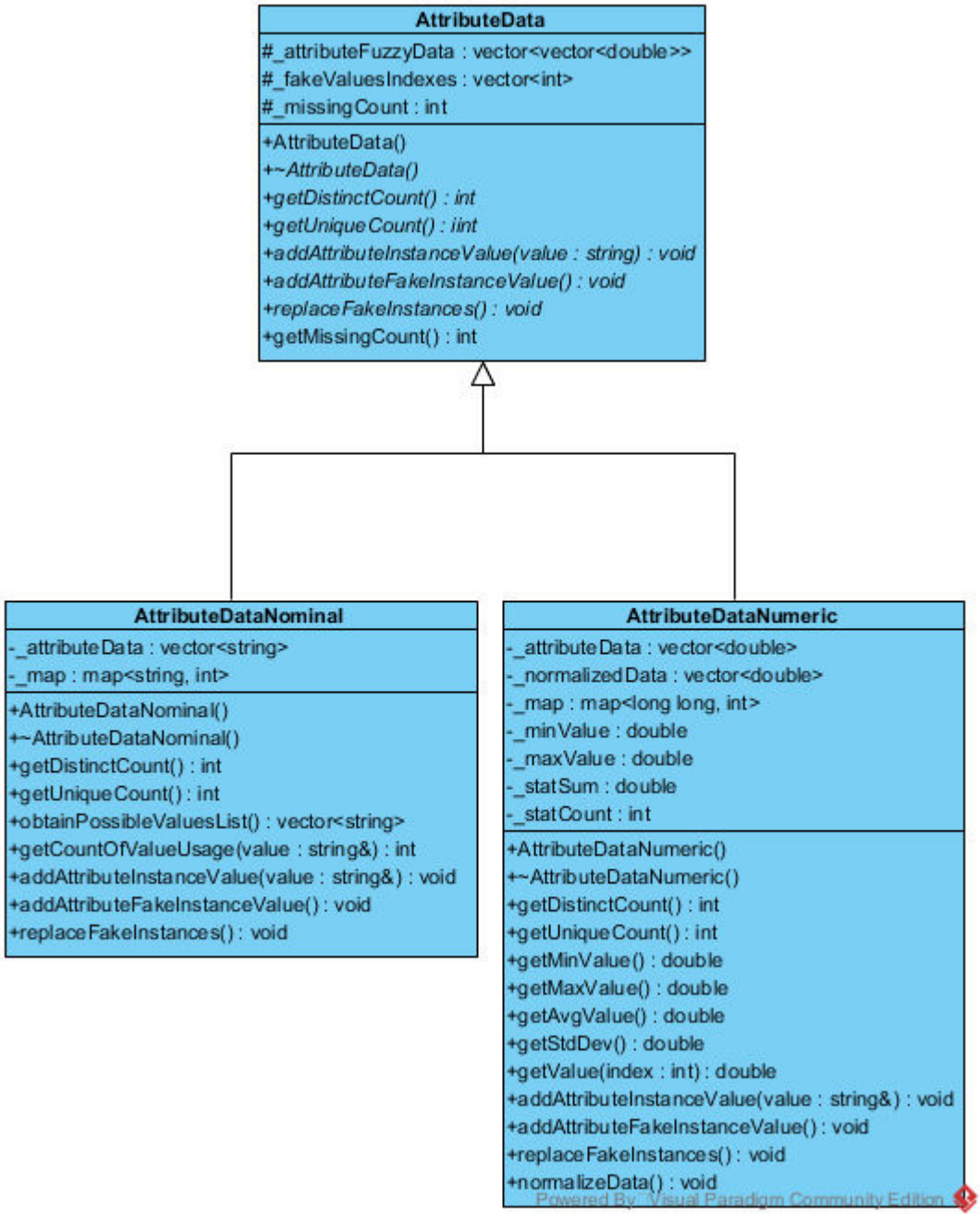


Figure 3.11. An AttributeData abstract class with derived classes.

The AttributeDataNominal class, that is also shown in Figure 3.11, has two own field, that are ***_attributeData*** (list of nominal values of the attribute) and ***_map*** (map container in which the key is a nominal attribute value and the value is number of repeating of this value in the set). In addition to overriding virtual methods of the base class, the AttributeDataNominal class also defines ***obtainPossibleValuesList()*** method, which functionality is delegated from the *getPossibleValuesList()* method of the NominalAttribute

class, and ***getCountOfValueUsage()*** method, which functionality is delegated from the *getCountOfValueUsage()* method of the NominalAttribute class too.

The next significant class that should be described in this Section is a **Fuzzification** class, that is responsible for data set fuzzification (see Figure 3.12). It has one field ***_dataset***, that is a pointer to the analyzed data set, and following methods:

- ***fuzzificateDataset()*** – forces fuzzification of whole data set;
- ***fuzzificateNumericAttribute()*** – makes fuzzification of a given numeric attribute with a possibility of normalizing;
- ***fuzzificateNominalAttribute()*** – makes fuzzification of a given nominal attribute.

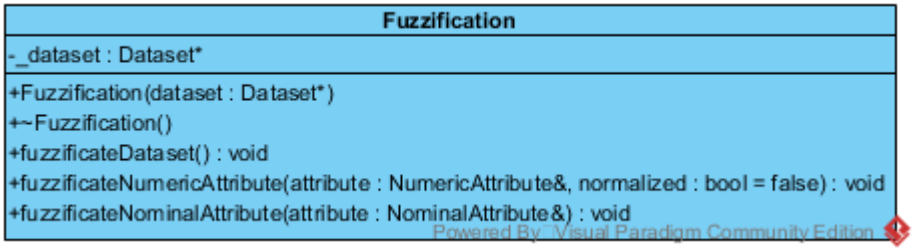


Figure 3.12. A Fuzzification class view.

A **FuzzyClusteringAlgorithm** class is a base class for each algorithm implemented in the tool (see Figure 3.13). It has one protected field ***_dataset***, which is a pointer to the analyzed data set, and one pure virtual method ***makeClustering()***, which runs fuzzy clustering algorithm.

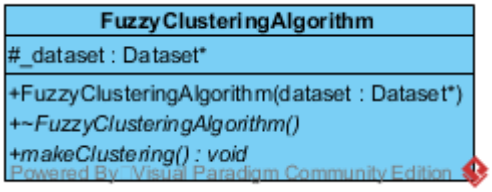


Figure 3.13. A FuzzyClusteringAlgorithm abstract class view.

The Fuzzy c-Means algorithm and its modifications Gustafson-Kessel and Gath-Geva algorithms, which were described in Section 2.2, are implemented in **FCM** **GustafsonKesselClustering** and **GathGevaClustring** classes correspondingly (see Figure 3.14).

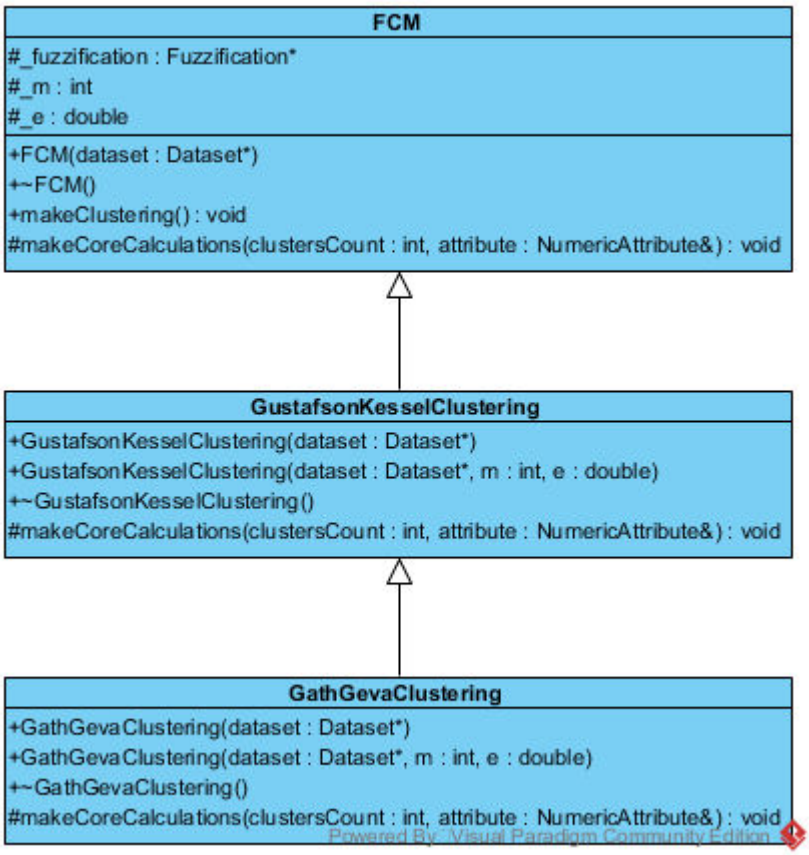


Figure 3.14. Classes, that implement FCM and its modifications.

The FCM class has three protected fields, that are ***_fuzzification*** (a pointer to object Fuzzification to provide the ability of calculating a pattern matrix), ***_m*** (a fuzzifier value), ***_e*** (a threshold – small positive number). The ***makeClustering()*** method overrides the pure virtual method from the base FuzzyClusteringAlgorithm class. The functionality after overriding is following: iterate all possible values of number of clusters, for each of them make fuzzy clustering and compute the pairing frequency CVI (see Section 2.2.4) and after iterating define the optimal number of clusters. Performing a fuzzy clustering on each iteration is done by invoking the ***makeCoreCalculations()*** method, which process all steps

of an algorithm. Thus, implementation of any modified FCM clustering algorithms requires only overriding of this method, that is the reason the GustafsonKesselClustering and the GathGevaClustring classes in Figure 3.14 are depicted with only one virtual protected method. Mentioned classes also contain additional private methods to make implementation of algorithms more readable, but it is not necessary to describe them here, because all their functionality can be moved to the *makeCoreCalculations()* method without any impact on functionality or design of the application.

A **MID** class, that is shown in Figure3.15, is an implementation of the Multi-Interval Discretization algorithm. It has only one own constant static field ***KERNEL_ACCURACY***, which equals to the denominator of the kernel relation $\Delta T/100$ (see Section 2.3), and overrides the ***makeClustering()*** method, where the algorithm steps are implemented.

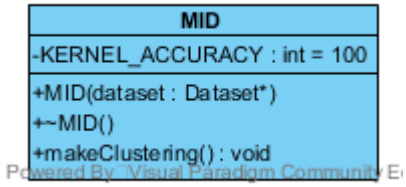


Figure 3.15. A MID class view.

A **FEBFC** class, shown in Figure 3.16, is an implementation of the FEBFC algorithm, described in Section 2.4. The class derives from the FuzzyClusteringAlgorithm class and additionally contains two protected fields: ***_validityIndexValue*** – a map, where *key* is an attribute index, and *value* is a total fuzzy entropy for this attribute; ***_fuzzification*** – a pointer to object Fuzzification for classification purposes.

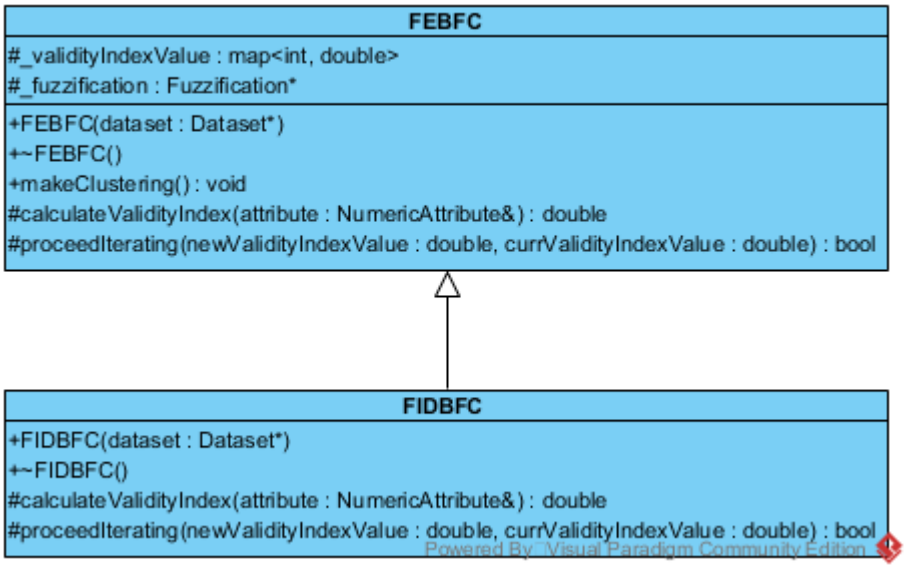


Figure 3.16. Classes, implementing FEBFC algorithm and its modification FIDBFC.

The FEBFC class has several methods implemented:

- ***makeClustering()*** – overrides a corresponding virtual method from the base class; purpose of this implementation is processing of all steps of the FEBFC algorithm;
- ***calculateValidityIndex()*** – this method is invoked from the *makeClustering()* method; it calculates a total fuzzy entropy of an attribute;
- ***proceedIterating()*** – this method is invoked from the *makeClustering()* method; it compares current value of a total fuzzy entropy with a previous one and return a decision: does it make sense to proceed iterating on not.

The *calculateValidityIndex()* and *proceedIterating()* methods are both virtual to allow overriding their functionality in the modifications of the FEBFC algorithm. One of the possible modifications, the FIDBFC algorithm, was proposed in this thesis in Section 2.5. Implementation of this algorithm was realized in a derived **FIDBFC** class (see Figure 3.16). This class uses *_validityIndexValue* field of its base class to store total fuzzy information density values of attribute instead of total fuzzy entropy. The *calculateValidityIndex()* and *proceedIterating()* methods are overridden to provide calculation and evaluation of the total fuzzy information density instead of total fuzzy entropy correspondingly.

As was mentioned in Section 3.1, the software tool must provide an ability of importing clustering results from external software. To cover this requirement a **FuzzyClustersImporter** class was implemented (see Figure 3.17).

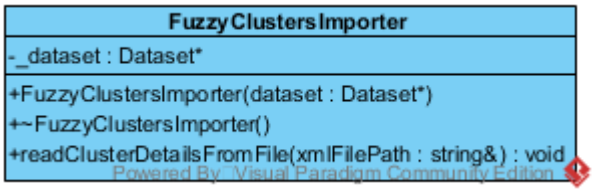


Figure 3.17. A FuzzyClustersImporter class view.

The FuzzyClustersImporter class completes all required functionality of the core module. It has one field (***_dataset*** – a pointer to the analyzed data set) and one public method (***readClusterDetailsFromFile()*** – reads information about the number of clusters and clusters borders for each attribute from *.xml* file and also calculates clusters centers to convert hard clusters into fuzzy clusters).

3.3. Graphical user interface implementation

The GUI module of the Fuzzy Clustering Tool was also implemented in C++ programming language using Qt 5.10 framework. It was chosen because Qt is probably the most popular cross-platform application framework for C++, which opens a possibility of easy moving the application from Windows to Linux or Mac OS.

The module was developed inside of the FuzzyClusteringTool "solution" in Visual Studio 2017 as a separate project **QtGuiFuzzyClustering** (see Figure 3.2).

This Section describes implementation of graphical user interface showing snapshots of the running application rather than demonstrating code aspects, because the module's goal is graphical interaction between a user and the application, so *look and feel* is more important

than *code behind*. In addition, any interested reader of this thesis, can familiarize with code of the Fuzzy Clustering Tool by himself, because it is enclosed to the diploma thesis on CD as an Appendix 2.

In Figure 3.18 the main window of the Fuzzy Clustering Tool is shown. It consists of six sections:

1. a menu bar, where user can choose a desired action;
2. a brief information about a loaded data set;
3. a list of attributes of the data set;
4. a detail information about a selected attribute;
5. a result of clustering;
6. a visualization of fuzzy clustering of the selected attribute.

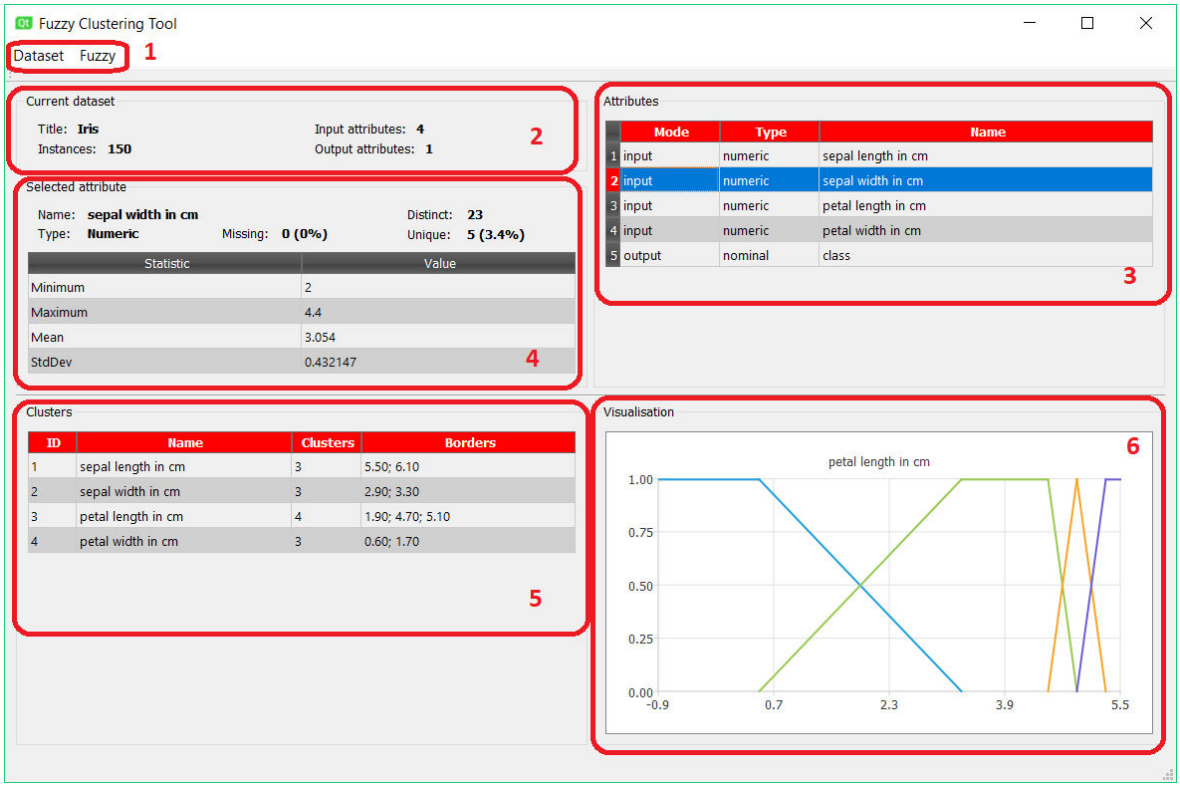


Figure 3.18. The main window of the Fuzzy Clustering Tool.

The menu bar has two high-level options, that are ***Dataset*** and ***Fuzzy***, as shown in Figure 3.18. If a user clicks on *Dataset* option, a menu with ***Open...*** and ***Exit*** actions is shown (see Figure 3.19).

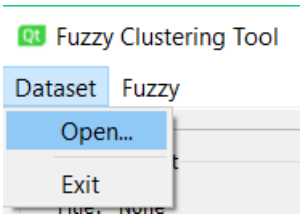


Figure 3.19. The *Dataset* menu.

Clicking on *Open...* action leads to opening a dialog window for selecting an .xml file, describing a data set to be loaded (opened). Clicking on *Exit* action leads to shutting down the application and closing the main window.

The other option that is located in the menu bar is *Fuzzy*. If the user clicks on it, the following menu is displayed (see Figure 3.20).

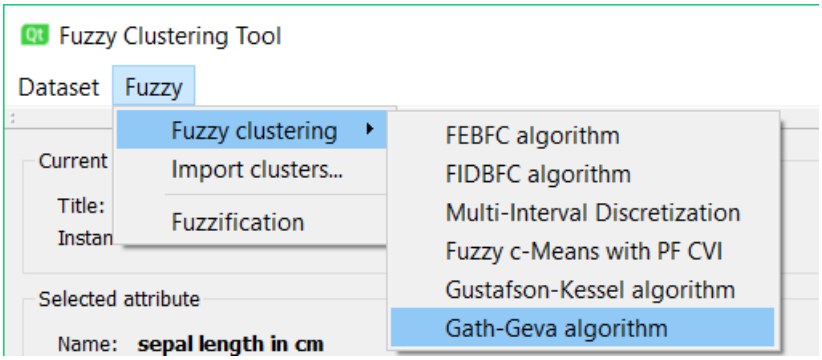


Figure 3.20. The *Fuzzy* menu.

In the *Fuzzy* menu the user can select one of the fuzzy clustering algorithms by clicking on its name, import clustering results from external .xml file by clicking on *Import clusters...* action or make fuzzification by clicking on action with an appropriate label.

The section with a brief information about a loaded data set (see Figure 3.21) displays to the user following information about the data set, that was opened by clicking on *Dataset/Open...*: the name of the opened data set (**Title**), the total number of patterns in the dataset (**Instances**), the number of input attributes (**Input attributes**) and the number of output attributes (**Output attributes**).

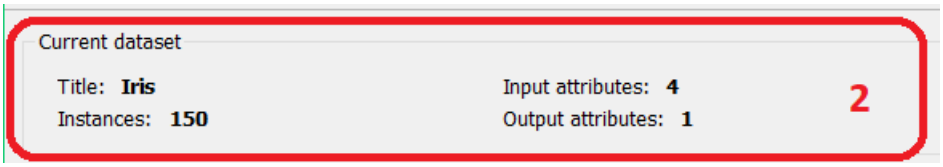


Figure 3.21. "Current dataset" section of the main window of the tool.

The section with a list of attributes of the data set is made in a form of table, where columns indicates an id, a mode, a type and a name of each attribute (see Figure 3.22).

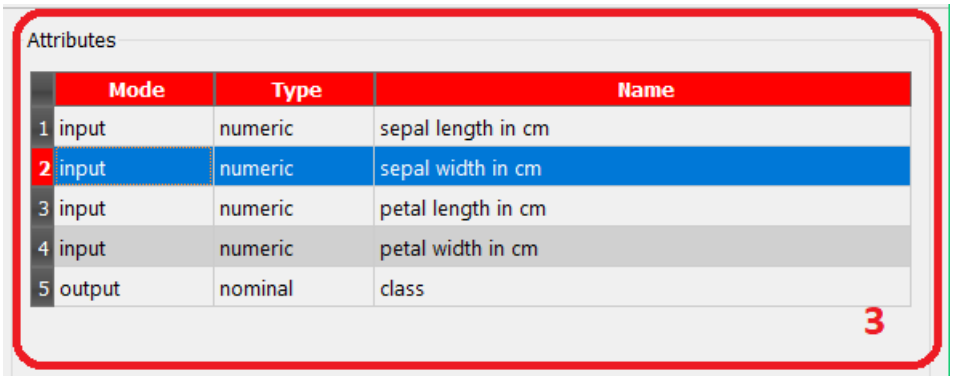


Figure 3.22. "Attributes" section of the main window of the tool.

If the user clicks on some attribute in the table above, the detailed information about it will be displayed in the section "Selected attribute" (see Figure 3.23). There is such information as minimum value of the attribute, maximum value, standard deviation, the number of missing values etc.

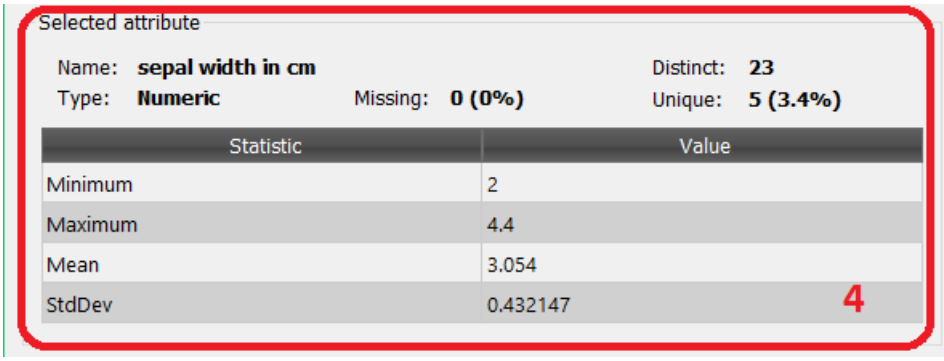


Figure 3.23. "Selected attribute" section of the main window of the tool.

After processing a clustering algorithm or importing a clustering result, the table of the clustering result will be shown (see Figure 3.24). In this table the information about the number of clusters and cluster borders can be found.

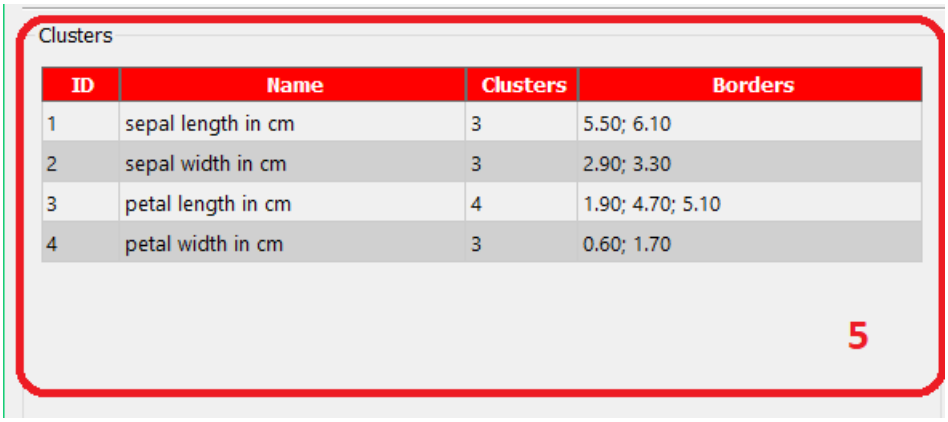


Figure 3.24. “Clusters” section of the main window of the tool.

Clicking on any row in the table in “Clusters” section leads to visualizing a fuzzy clustering result of selected attribute on a chart (see Figure 3.25).

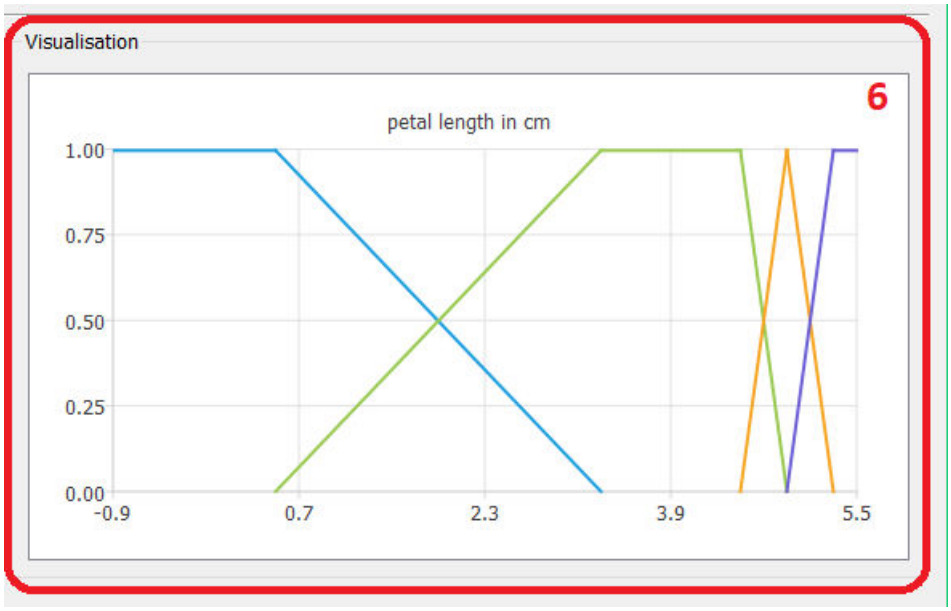


Figure 3.25. “Visualization” section of the main window of the tool.

CHAPTER 4. EXPERIMENTAL STUDY WITH THE IMPLEMENTED SOLUTION

4.1. Fuzzy clustering accuracy evaluation

After processing a clustering algorithm, the optimal number of clusters becomes known as well as the partition matrix. But each algorithm may result in different partitioning of the data set, i.e. different algorithms may return different values of the optimal number of clusters or different partition matrices. In this case we need to determine, which clustering result is accurate.

Evaluation of fuzzy clustering results can be made through the Clustering Accuracy Indices, that can be divided into two groups: Internal indices (uses only input attributes data) and External indices (uses information about belonging of a pattern to some class of data) [31]. The Internal indices can be used in both supervised and unsupervised learning, but the External indices can be used in case of supervised learning only.

In a literature of clustering a lot of various internal indices can be found, but the most well-known of them are the following [32]:

- Partition Coefficient index;
- Partition Entropy index;
- Fukuyama-Sugeno index;
- Xie-Beni index.

The listed above indices were already mentioned in Section 2.1 in the context of determining an optimal number of clusters. But the field of their usage is not limited by this context, they can also be used for comparing of clustering results, obtained with different algorithms. For comparing the index values with the aim of defining the most qualitative clustering result, the appropriate optimality criterions, mentioned in Section 2.1, can be used.

Among the external indices the following can be highlighted [33, 34]:

- Purity index;
- Normalized Mutual Information index.

Both of above external indices are based on contingency matrix, that can be defined using following formula:

$$V = \begin{pmatrix} v_{11} & v_{12} & \dots & v_{1c} \\ v_{21} & v_{22} & \dots & v_{2c} \\ \dots & \dots & \ddots & \dots \\ v_{k1} & v_{k2} & \dots & v_{kc} \end{pmatrix} \tag{4.1}$$

where v_{ij} is a sum of fuzzy values of an attribute, which are labeled to the j-th class and belong to the i-th cluster simultaneously; c is the number of classes of the data set; k is the number of clusters into which the attribute is divided.

The Purity index (I_{purity}) is a transparent measure for clustering result evaluation calculated as:

$$I_{purity} = \frac{1}{n}\sum_{i=1}^{k} \max_{1 \le j \le c}(v_{ij})$$

where n – the number of patterns in the data set; $0 \le I_{purity} \le 1$. The closer the value of the Purity index to 1, the better the result of clustering.

The Normalized Mutual Information (NMI) index can be calculated according to the following formula:

$$I_{NMI} = \frac{I(K;C)}{\sqrt{H(K) \cdot H(C)}}$$

where K – the set of clusters; C – the set of classes; $I(K;C)$ – the mutual information, $I(K;C) = \sum_{i=1}^{k}\sum_{j=1}^{c}\left(\frac{v_{ij}}{n}\right)\log\left(\frac{n \cdot v_{ij}}{\sum_{i=1}^{k} v_{ij} \cdot \sum_{j=1}^{c} v_{ij}}\right)$; $H(K)$ and $H(C)$ are entropies of cluster and classes correspondingly $H(K) = \sum_{i=1}^{k}\frac{\sum_{j=1}^{c} v_{ij}}{n}\log\frac{\sum_{j=1}^{c} v_{ij}}{n}$; $H(C) = \sum_{j=1}^{c}\frac{\sum_{i=1}^{k} v_{ij}}{n}\log\frac{\sum_{i=1}^{k} v_{ij}}{n}$.

In the next Section the clustering algorithms, described in Chapter 2, are evaluated and compared using mentioned above internal and external indices calculated for the following medical data sets: Pima Indians Diabetes, Heart Disease, Breast Cancer Wisconsin, Indian Liver Patient Records and Chronic Kidney Disease.

4.2. Comparison of the fuzzy clustering algorithms on medical data

4.2.1. Pima Indians Diabetes

A data set of the Pima Indians Diabetes data set was taken from the Kaggle repository. This data set can be used in machine learning to construct a prediction system which would identify whether a patient has diabetes depending on certain diagnostic measurements [35].

The data set contains *768 instances*; *8 input attributes* and *1 output*. The attributes of the Pima Indians Diabetes are described in the Table 4.1.

Table 4.1. Attributes of the Pima Indians Diabetes data set: **A_1-A_8** are input attributes and **C** is an output attribute

Attribute	Type	Description	Possible nominal values	Min. value	Max. value	Mean value	Standard deviation
A_1	numeric	Number of times pregnant	—	0	17	3.845	3.367
A_2	numeric	Plasma glucose concentration a 2 hours in an oral glucose tolerance test	—	0	199	120.895	31.952
A_3	numeric	Diastolic blood pressure (mm Hg)	—	0	122	69.106	19.343
A_4	numeric	Triceps skin fold thickness (mm)	—	0	99	20.537	15.942
A_5	numeric	2-Hour serum insulin (mu U/ml)	—	0	846	79.799	115.169
A_6	numeric	Body mass index (weight in kg/(height in m)^2)	—	0	67.1	31.993	7.879
A_7	numeric	Diabetes pedigree function	—	0.078	2.42	0.472	0.331
A_8	numeric	Age (years)	—	21	81	33.241	11.753
C	nominal	Outcome	*0* – a patient does not have diabetes; *1* – a patient has diabetes	—	—	—	—

As a result of performing fuzzy clustering on the Pima Indians Diabetes data set using the algorithms, described in Chapter 2, the membership functions, shown in Figures 4.1-4.6, were obtained.

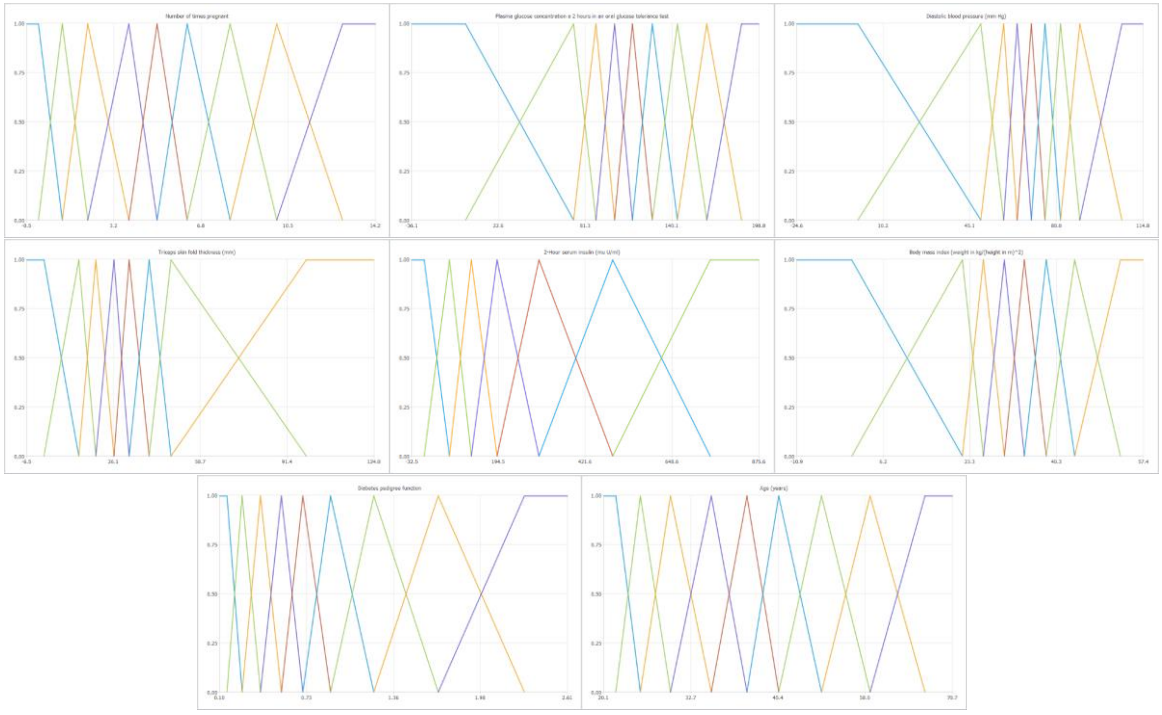


Figure 4.1. Membership functions of attributes **A1** – **A8** obtained using the FCM algorithm with Pairing Frequency index for getting the optimal number of clusters

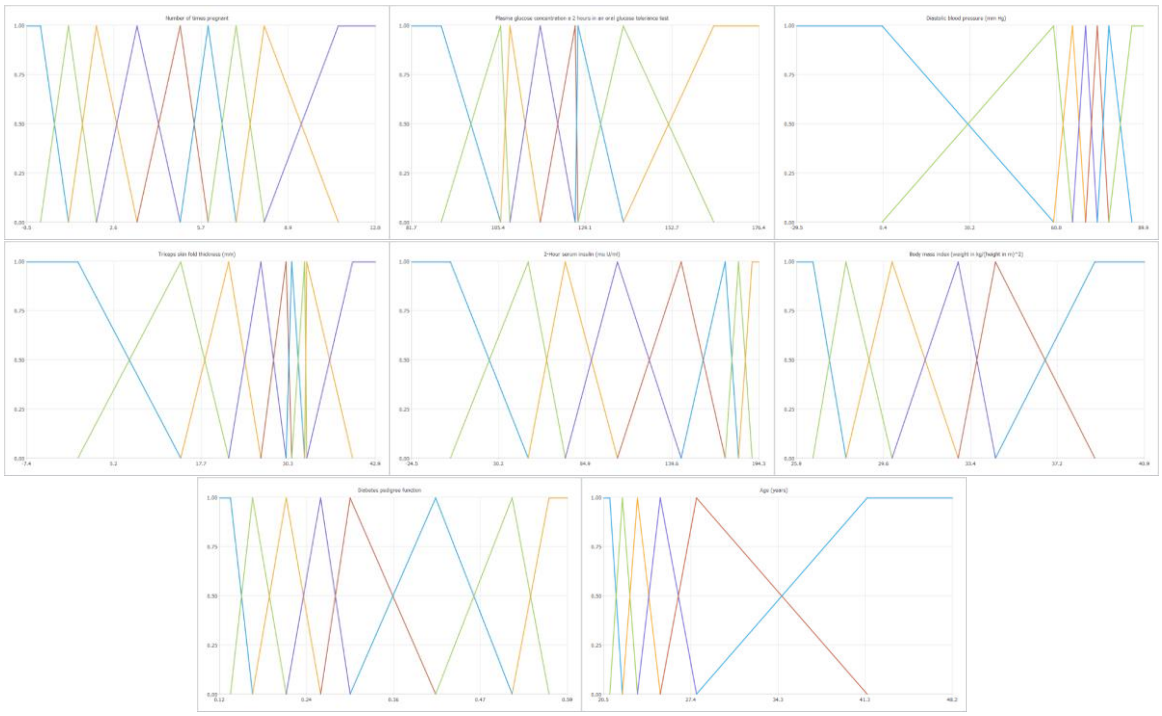


Figure 4.2. Membership functions of attributes **A1** – **A8** obtained using the GK algorithm with Pairing Frequency index for getting the optimal number of clusters

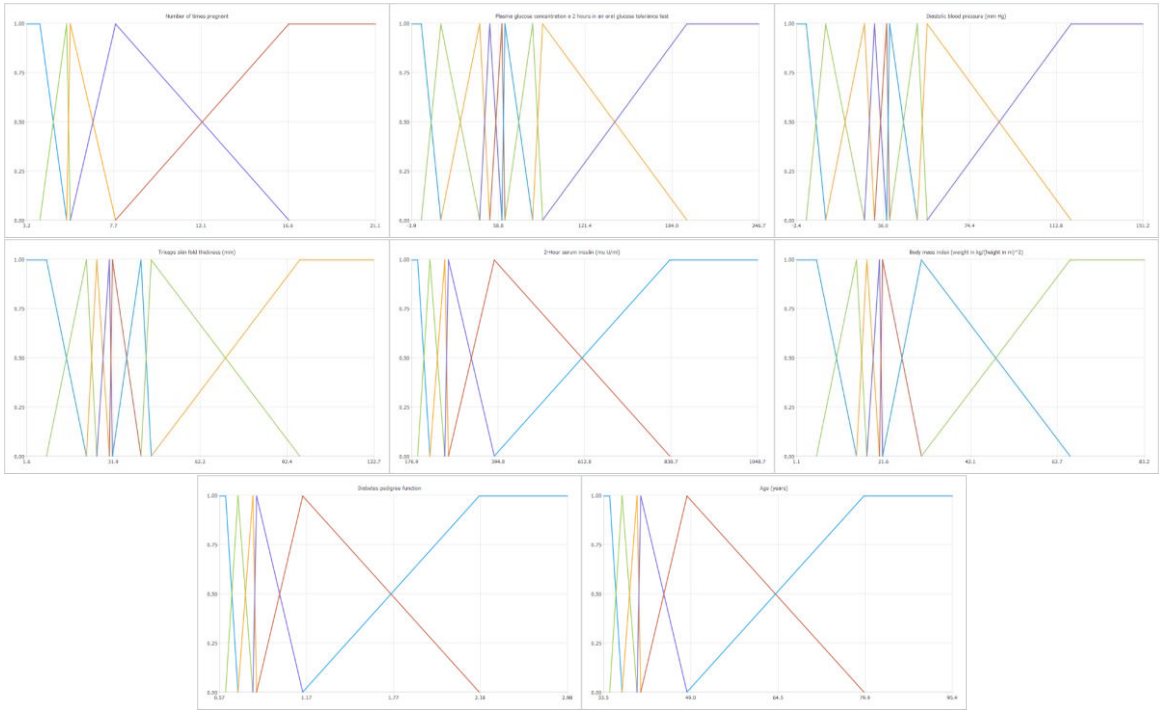


Figure 4.3. Membership functions of attributes $\mathbf{A_1}$ – $\mathbf{A_8}$ obtained using the GG algorithm with Pairing Frequency index for getting the optimal number of clusters

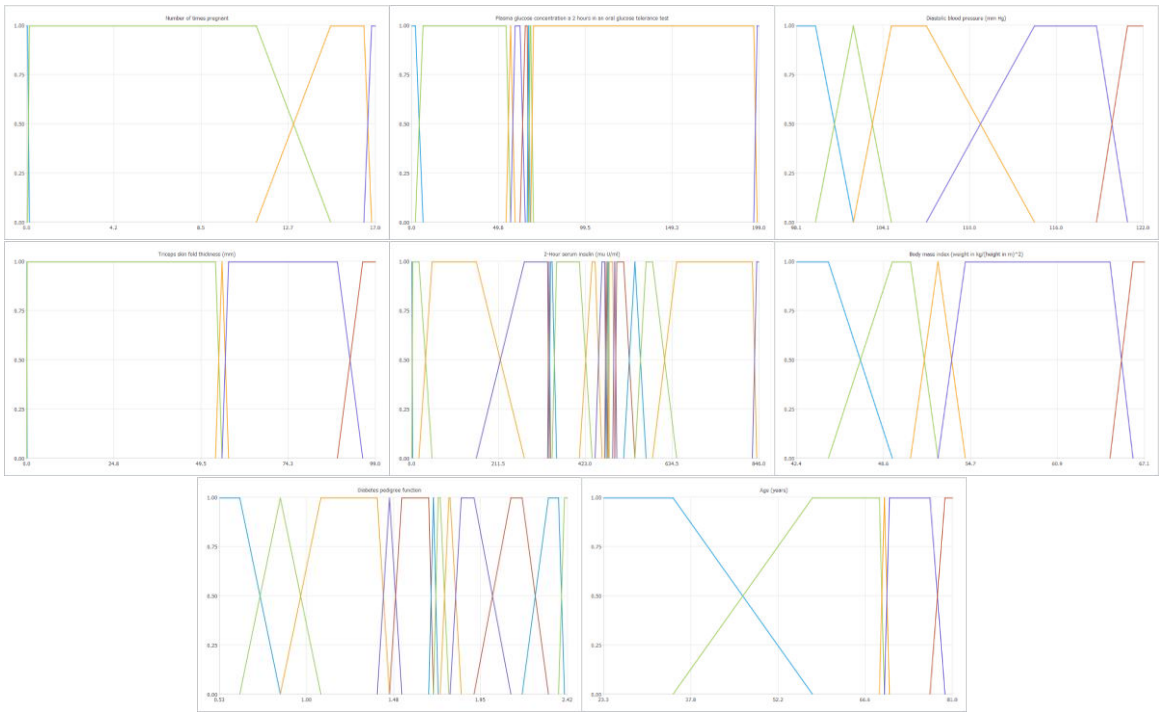


Figure 4.4. Membership functions of attributes $\mathbf{A_1}$ – $\mathbf{A_8}$ obtained using the MID algorithm

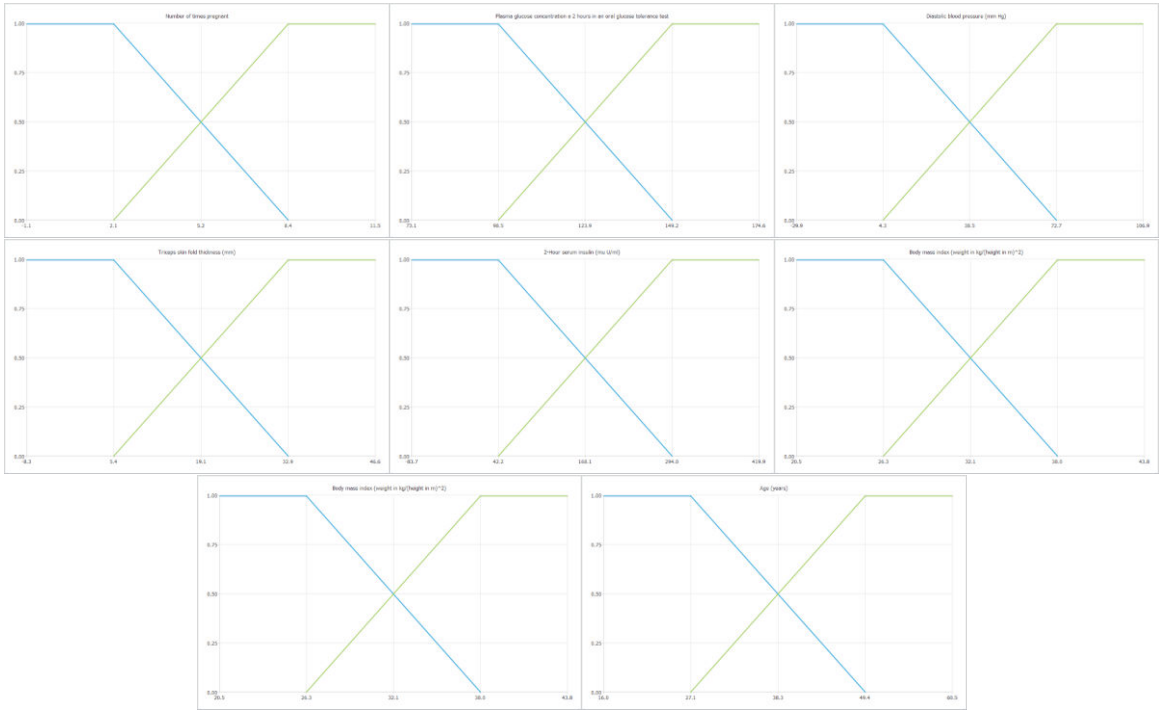


Figure 4.5. Membership functions of attributes $\mathbf{A_1}$ – $\mathbf{A_8}$ obtained using the FEBFC algorithm

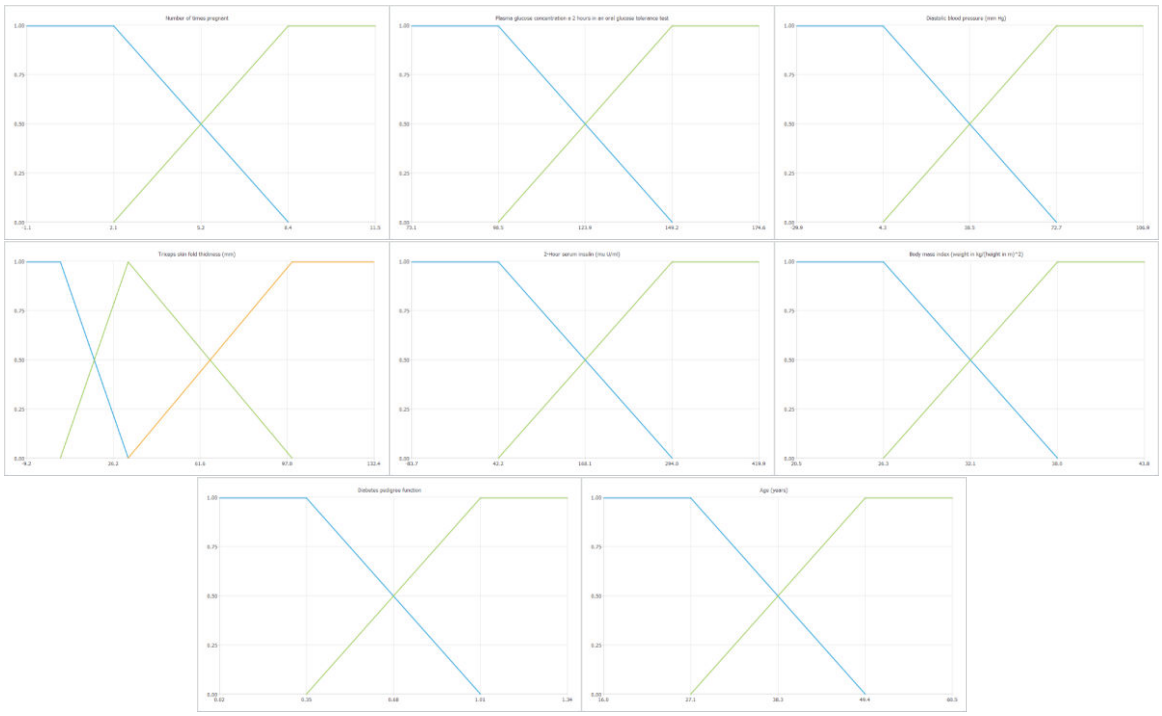


Figure 4.6. Membership functions of attributes $\mathbf{A_1}$ – $\mathbf{A_8}$ obtained using the FIDBFC algorithm

Based on the obtained membership functions, fuzzification of the data set was made for transforming crisp numerical values into fuzzy values. Then the evaluation indices, described in the Section 4.1, were calculated to compare clustering accuracy of the algorithms (see Table 4.2).

Table 4.2. Clustering Accuracy Indices calculated for the fuzzification performed on the Pima Indians Diabetes Dataset

Algorithm	Partition Coefficient index	Partition Entropy index	Fukuyama-Sugeno index	Xie-Beni index	Purity index	Normalized Mutual Information index
FCM	0.74530	0.55755	15.49595	0.13241	0.67337	0.04168
GK	0.81605	0.39656	9.10605	62.95703	0.66701	0.04180
GG	0.80680	0.42239	7.03228	129.22207	0.66321	0.03090
MID	0.96787	0.06993	9.38145	187.26315	0.65865	0.02023
FEBFC	0.84070	0.34661	5.35528	0.09274	0.65654	0.02903
FIDBFC	0.83577	0.35898	5.69637	0.09239	0.65654	0.02913

According to the obtained values of the Partition Coefficient and the Partition Entropy indices, the most accurate is the MID algorithm. But on the other hand, this algorithm gives us the worst membership functions of some attributes (see Figure 4.3). To understand this contradiction we should consider, that the mentioned indices are the most primitive in taking into account different characteristics of a partition. In this context, more relevant is the second result, the FEBFC algorithm, which clustering results is close to the original set of classes of the data set. The FEBFC algorithm is the most accurate according to the Fukuyama-Sugeno index. The FIDBFC algorithm is the most accurate according to the Xie-Beni index. But in general, among all indices, the results of the FIDBFC are quite similar to the results of the FEBFC. The Purity and Normalized Mutual Information indices identify the FCM as the most accurate algorithm.

Therefore, most of the indices shows that the FEBFC and FIDBFC algorithms are the most accurate for the Pima Indians Diabetes Dataset and the FIDBFC, which is the modification of the FEBFC, mentioned in Section 2.5, gives better results than the FEBFC according to the Xie-Beni index.

4.2.2. Heart Disease

The Heart Disease data sets are located in the UCI machine learning repository [36]. They contain data from the following locations: Cleveland Clinic Foundation, Hungarian Institute of Cardiology (Budapest), V.A. Medical Center (Long Beach, CA), University Hospital (Zurich, Switzerland). In this thesis data from Cleveland Clinic Foundation are used only.

The Cleveland data set contains *303 patterns*, that were originally represented by 76 attributes. David W. Aha, who uploaded this data set to UCI repository, extracted 14 the most significant of them (*13 input attributes* and *1 output attribute*), which are described in Table 4.3.

Table 4.3. Attributes of the Cleveland Heart Disease data set: **A_1-A_{13}** are input attributes and **C** is an output attribute

Attribute	Type	Description	Possible nominal values	Min. value	Max. value	Mean value	Standard deviation
A_1	numeric	Age of a patient in years	—	29	77	54.439	9.024
A_2	nominal	Sex of a patient	*0* – female; *1* – male	—	—	—	—
A_3	nominal	Chest pain type	*1* – typical angina; *2* – atypical angina; *3* – non-anginal pain; *4* – asymptomatic	—	—	—	—
A_4	numeric	Resting blood pressure (in mm Hg on admission to the hospital)	—	94	200	131.690	17.571
A_5	numeric	Serum cholesterol in mg/dl	—	126	564	246.693	51.691
A_6	nominal	Fasting blood sugar > 120 mg/dl	*0* – false; *1* – true	—	—	—	—
A_7	nominal	Resting electrocardiographic results	*0* – normal; *1* – having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV); *2* – showing probable or definite left ventricular hypertrophy by Estes' criteria	—	—	—	—
A_8	numeric	Maximum heart rate achieved	—	71	202	149.607	22.837
A_9	nominal	Exercise induced angina	*0* – no; *1* – yes	—	—	—	—

(continued)

Attribute	Type	Description	Possible nominal values	Min. value	Max. value	Mean value	Standard deviation
A_{10}	numeric	ST depression induced by exercise relative to rest	—	0	6.2	1.040	1.159
A_{11}	nominal	The slope of the peak exercise ST segment	*1* – upsloping; *2* – flat; 3 – downsloping	—	—	—	—
A_{12}	nominal	Number of major vessels colored by fluoroscopy	*0*; *1*; *2*; *3*	—	—	—	—
A_{13}	nominal	Thallium heart scan	*3* – normal; *6* – fixed defect; *7* – reversible defect	—	—		—
C	nominal	Diagnosis of heart disease	*0* – absent; *1* – class I of heart failure; *2* – class II of heart failure; *3* – class III of heart failure; *4* – class IV of heart failure	—	—	—	—

In a result of fuzzy clustering using the algorithms described in Chapter 2, membership functions were obtained for each numeric attribute of the Heart Disease data set. They are grouped by the corresponding algorithms and shown in Figures 4.7-4.12.

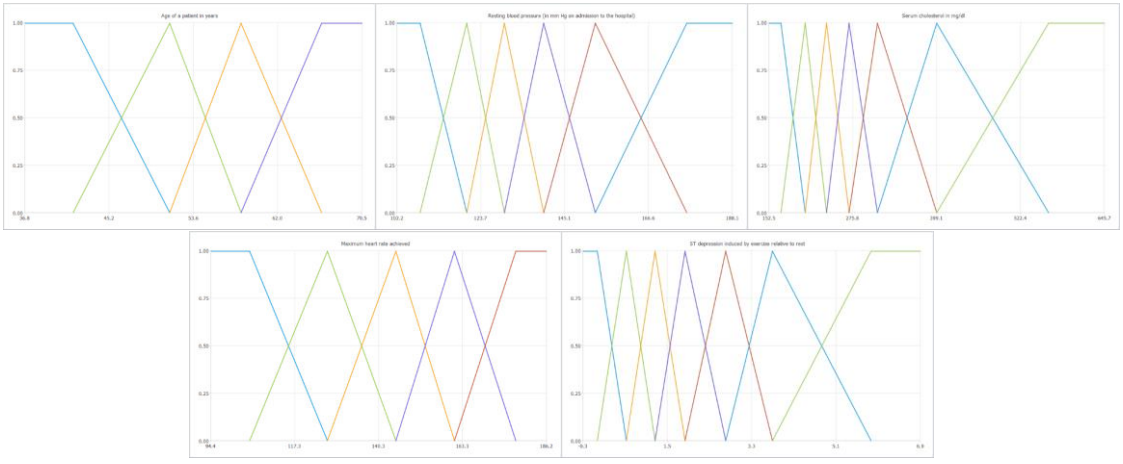


Figure 4.7. Membership functions of attributes **A_1**, **A_4**, **A_5**, **A_8** and **A_{10}** obtained using the FCM algorithm with Pairing Frequency index for getting the optimal number of clusters

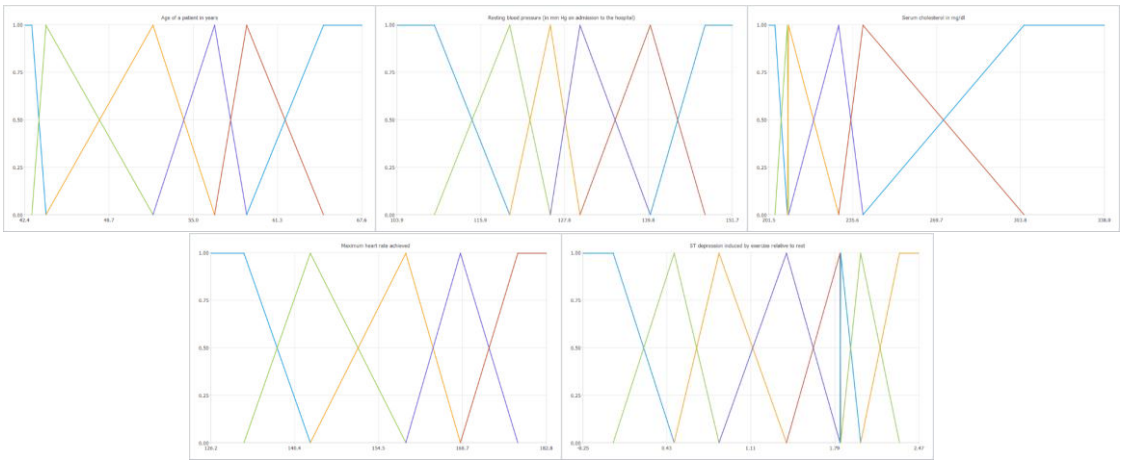


Figure 4.8. Membership functions of attributes $\mathbf{A_1}$, $\mathbf{A_4}$, $\mathbf{A_5}$, $\mathbf{A_8}$ and $\mathbf{A_{10}}$ obtained using the GK algorithm with Pairing Frequency index for getting the optimal number of clusters

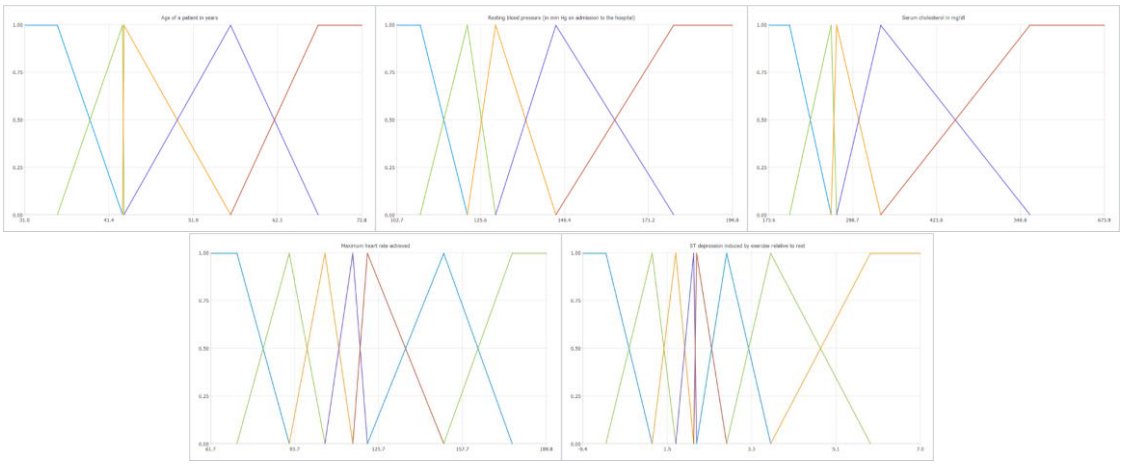


Figure 4.9. Membership functions of attributes $\mathbf{A_1}$, $\mathbf{A_4}$, $\mathbf{A_5}$, $\mathbf{A_8}$ and $\mathbf{A_{10}}$ obtained using the GG algorithm with Pairing Frequency index for getting the optimal number of clusters

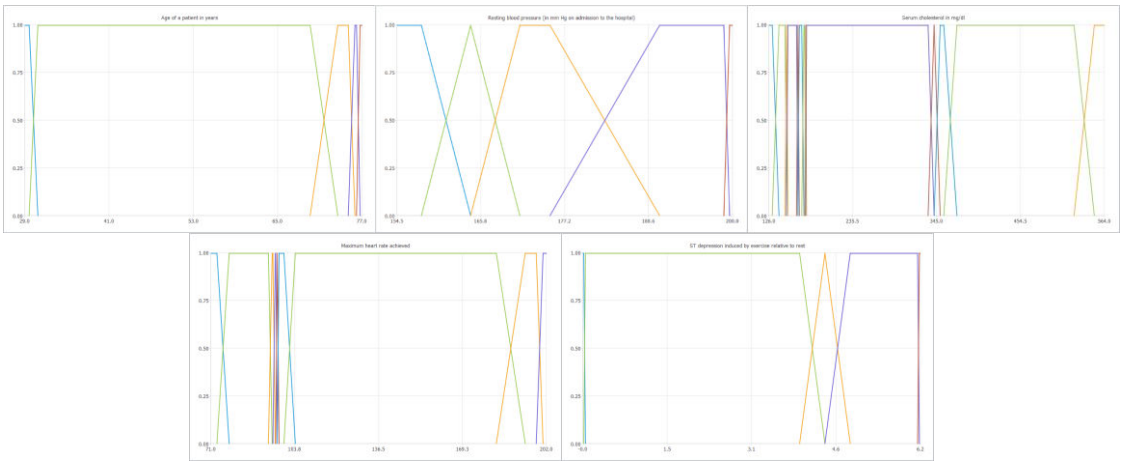


Figure 4.10. Membership functions of attributes $\mathbf{A_1}$, $\mathbf{A_4}$, $\mathbf{A_5}$, $\mathbf{A_8}$ and $\mathbf{A_{10}}$ obtained using the MID algorithm

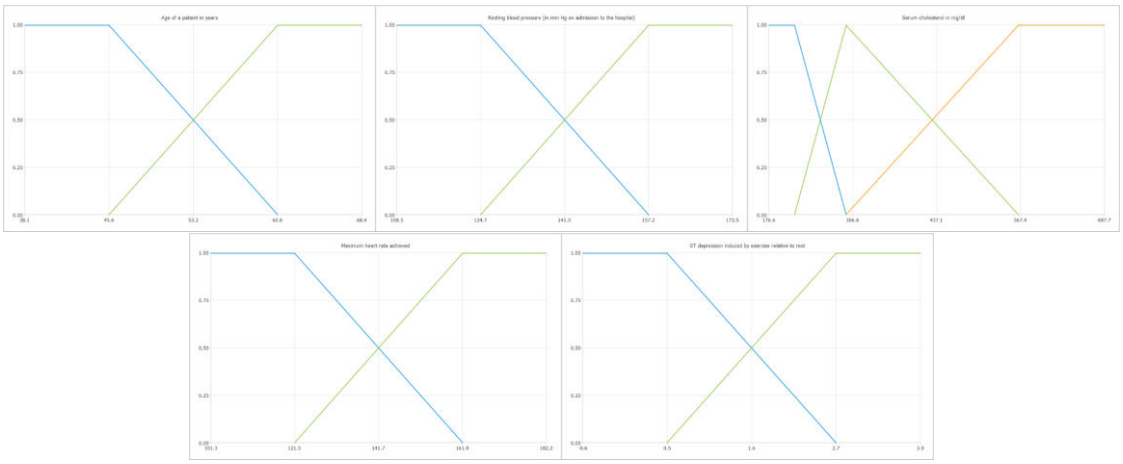


Figure 4.11. Membership functions of attributes $\mathbf{A_1}$, $\mathbf{A_4}$, $\mathbf{A_5}$, $\mathbf{A_8}$ and $\mathbf{A_{10}}$ obtained using the FEBFC algorithm

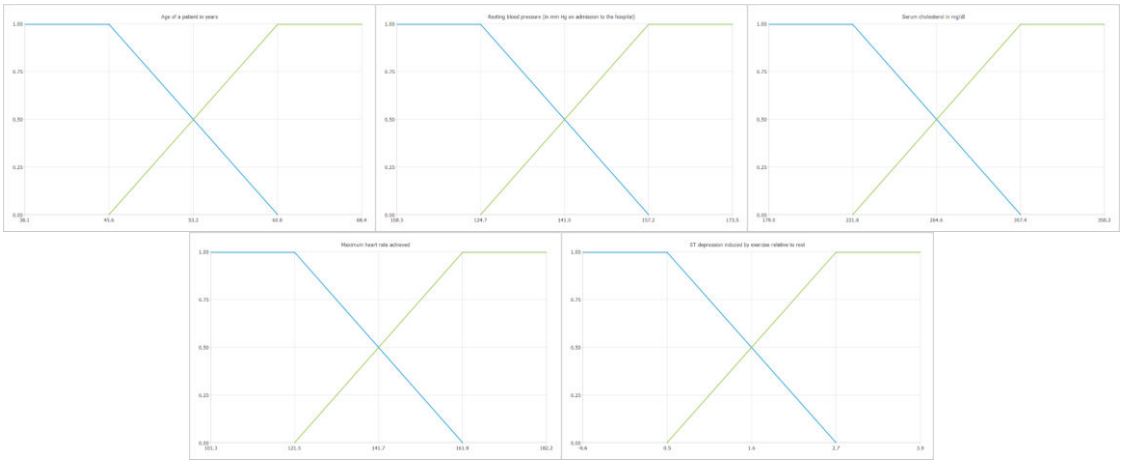


Figure 4.12. Membership functions of attributes $\mathbf{A_1}$, $\mathbf{A_4}$, $\mathbf{A_5}$, $\mathbf{A_8}$ and $\mathbf{A_{10}}$ obtained using the FIDBFC algorithm

Based on the obtained membership functions, fuzzification of the data set was made and the appropriate evaluation indices, described in the Section 4.1, were calculated. The results of calculation are shown in Table 4.4.

Table 4.4. Clustering Accuracy Indices calculated for the fuzzification performed on the Cleveland Heart Disease data set

Algorithm	Partition Coefficient index	Partition Entropy index	Fukuyama-Sugeno index	Xie-Beni index	Purity index	Normalized Mutual Information index
FCM	0.74495	0.56002	5.92440	0.11528	0.54916	0.04926
GK	0.80537	0.42032	3.17181	9.16700	0.54603	0.05179
GG	0.73040	0.58724	5.84767	62.60825	0.55168	0.04751
MID	0.98750	0.02764	4.50228	313.08871	0.54940	0.06126
FEBFC	0.81525	0.40006	2.25668	0.10192	0.54125	0.03940
FIDBFC	0.82118	0.38605	2.00213	0.10587	0.54125	0.03905

As we can see in Table 4.4, the MID algorithm gives the most accurate clustering results according to the Partition Coefficient, the Partition Entropy and the Normalized Mutual Information indices. Clustering results of the GG algorithm are the most accurate only according to the Purity index as well as clustering results of the FEBFC algorithm are the most accurate according to the Xie-Beni index.

The FIDBFC algorithm is the most accurate according to the Fukuyama-Sugeno index. It also better than the original FEBFC algorithm according to the Partition Coefficient and the Partition Entropy indices, has the same accuracy according to the Purity index and a bit worse according to other two indices. Thus, modification of the FEBFC algorithm, proposed in this thesis, leads to better clustering results of the Heart Disease data set according to at least half of considered indices.

4.2.3. Breast Cancer Wisconsin

The Breast Cancer Wisconsin data set was taken from the UCI machine learning repository [36]. The data set contains *569 patterns*, represented by 32 attributes, that were computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. In this experiment only 11 attributes (*10 input attributes* and *1 output*), that describe mean values of characteristics of the cell nuclei present in the image, are considered (see Table 4.5).

Table 4.5. Attributes of the Breast Cancer Wisconsin data set: **A_1-A_{10}** are input attributes and **C** is an output attribute

Attribute	Type	Description	Possible nominal values	Min. value	Max. value	Mean value	Standard deviation
A_1	numeric	Radius (mean of distances from center to points on the perimeter)	—	6.981	28.110	14.127	3.521
A_2	numeric	Texture (standard deviation of gray-scale values)	—	9.710	39.280	19.290	4.297
A_3	numeric	Perimeter	—	43.790	188.500	91.969	24.278
A_4	numeric	Area	—	143.500	2501	654.889	351.605
A_5	numeric	Smoothness (local variation in radius lengths)	—	0.053	0.163	0.096	0.014
A_6	numeric	Compactness (perimeter^2 / area - 1.0)	—	0.019	0.345	0.104	0.053
A_7	numeric	Concavity (severity of concave portions of the contour)	—	0	0.427	0.089	0.080
A_8	numeric	Concave points (number of concave portions of the contour)	—	0	0.201	0.049	0.039
A_9	numeric	Symmetry	—	0.106	0.304	0.181	0.027
A_{10}	numeric	Fractal dimension ("coastline approximation" - 1)	—	0.050	0.097	0.063	0.007
C	nominal	Diagnosis	*M* – malignant; *B* – benign	—	—	—	—

The fuzzy clustering algorithms described in Chapter 2 were performed on each numeric attribute of the Breast Cancer Wisconsin data set. As a result, the membership functions, shown in Figures 4.13-4.18 were obtained.

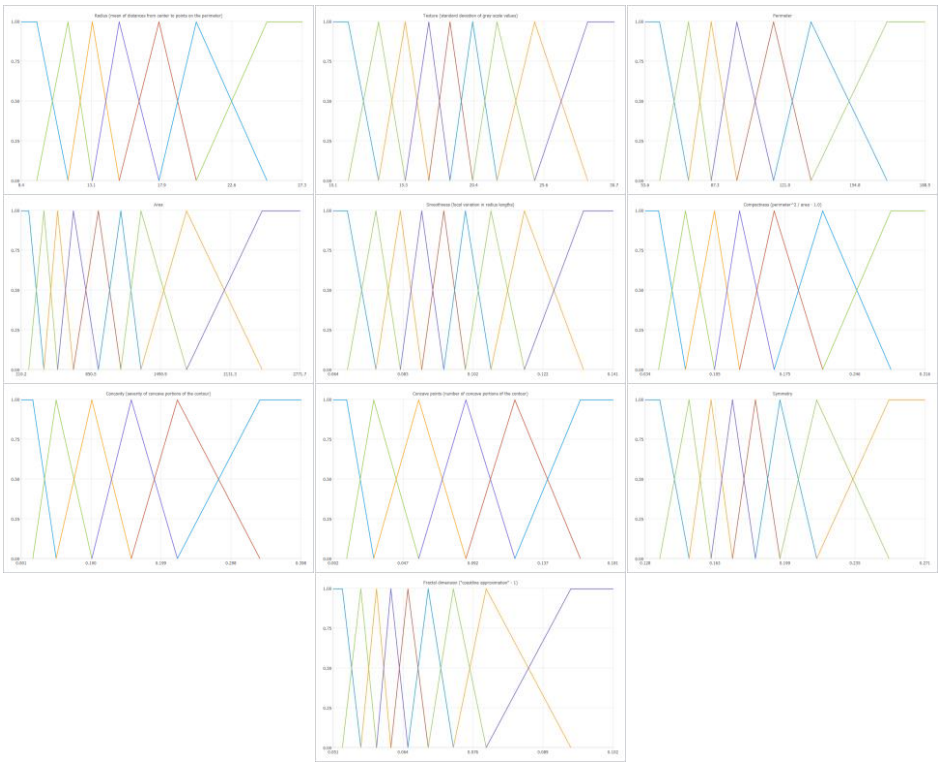


Figure 4.13. Membership functions of attributes $\mathbf{A_1}$ – $\mathbf{A_{10}}$ obtained using the FCM algorithm with Pairing Frequency index for getting the optimal number of clusters

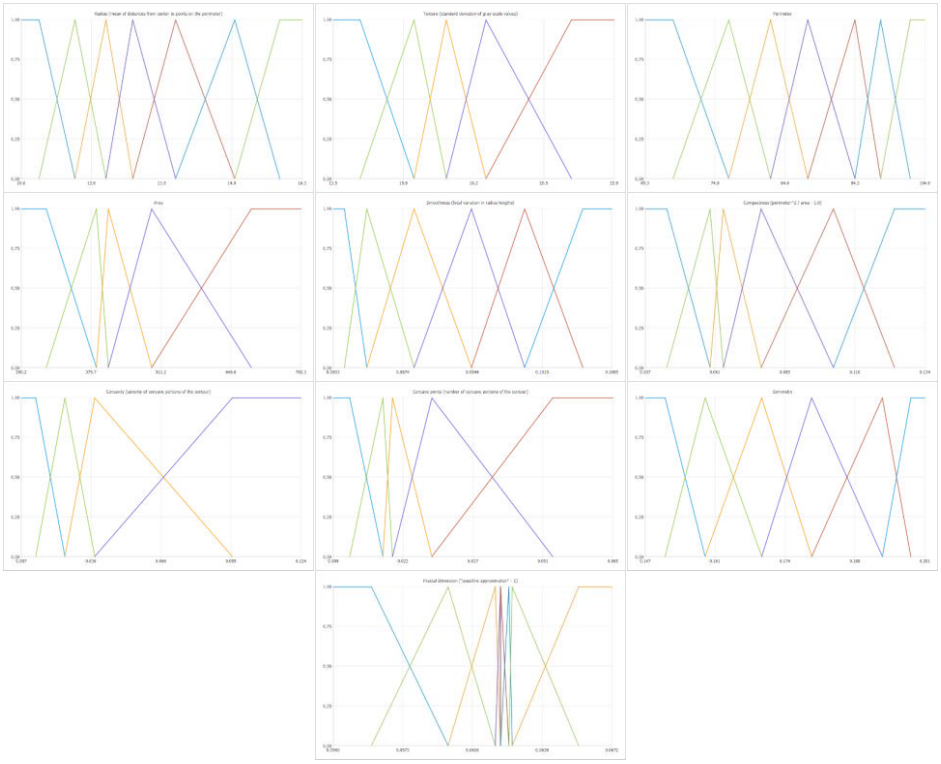


Figure 4.14. Membership functions of attributes $\mathbf{A_1}$ – $\mathbf{A_{10}}$ obtained using the GK algorithm with Pairing Frequency index for getting the optimal number of clusters

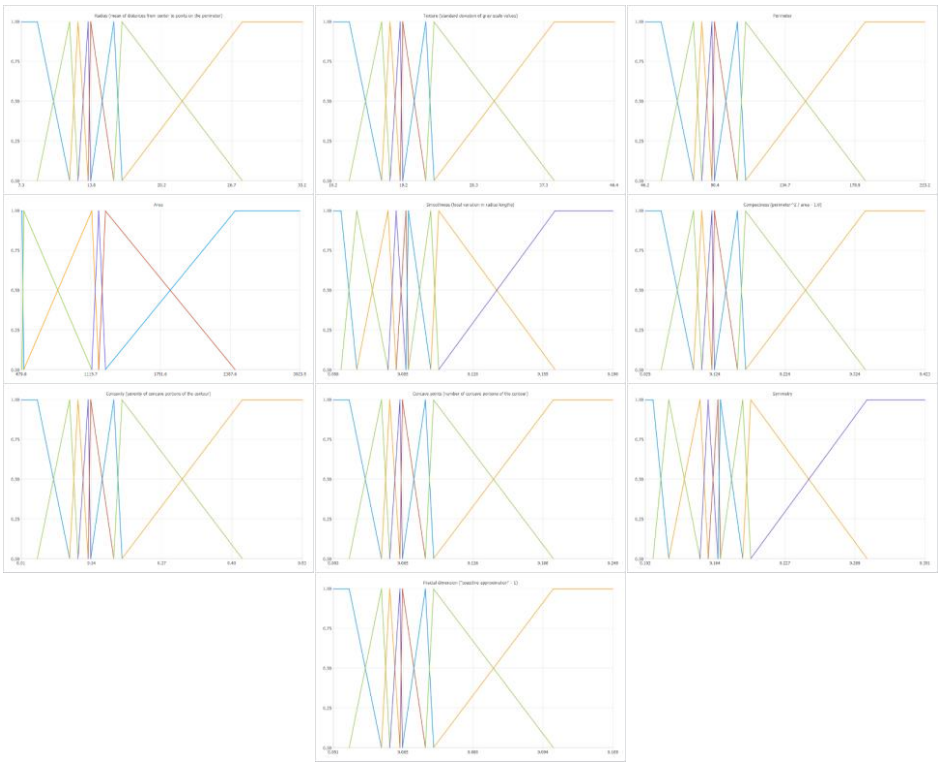


Figure 4.15. Membership functions of attributes $\mathbf{A_1}$ – $\mathbf{A_{10}}$ obtained using the GG algorithm with Pairing Frequency index for getting the optimal number of clusters

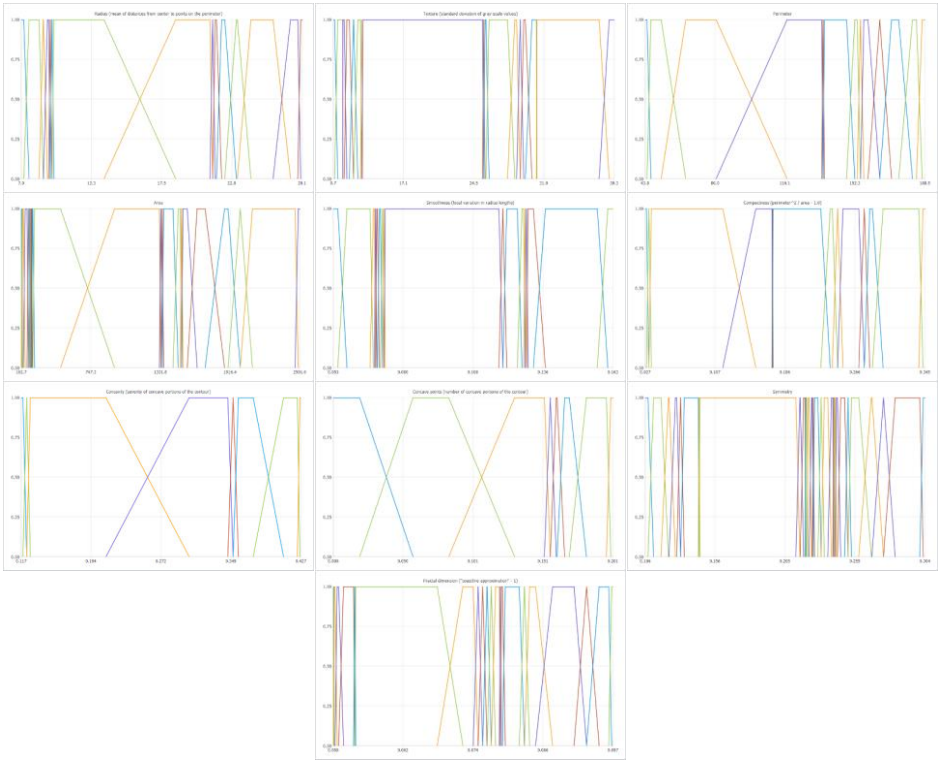


Figure 4.16. Membership functions of attributes $\mathbf{A_1}$ – $\mathbf{A_{10}}$ obtained using the MID algorithm

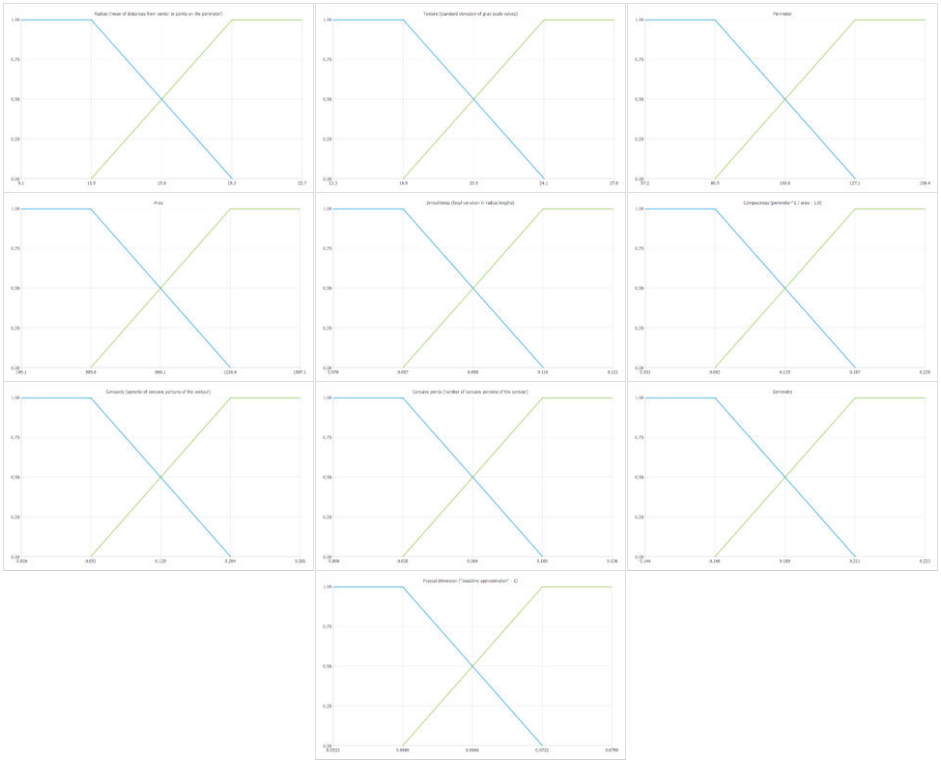


Figure 4.17. Membership functions of attributes $\mathbf{A_1}$ – $\mathbf{A_{10}}$ obtained using the FEBFC algorithm

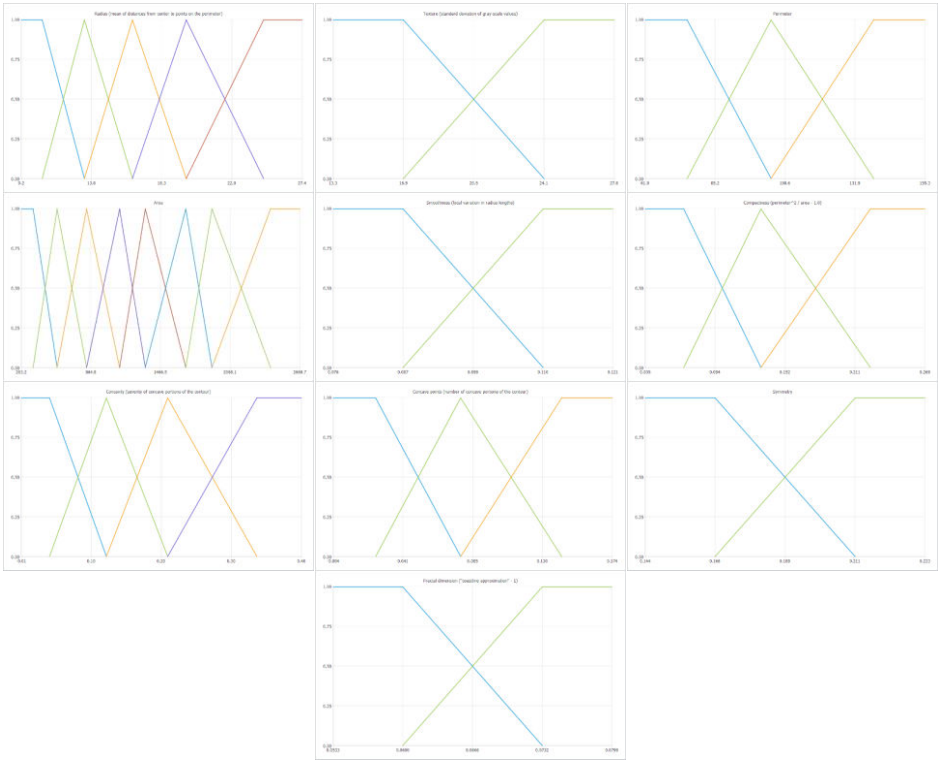


Figure 4.18. Membership functions of attributes $\mathbf{A_1}$ – $\mathbf{A_{10}}$ obtained using the FIDBFC algorithm

Based on the obtained membership functions, fuzzification of the data set was made and the appropriate evaluation indices, described in the Section 4.1, were calculated. The results of calculation are shown in Table 4.6.

Table 4.6. Clustering Accuracy Indices calculated for the fuzzification performed on the Breast Cancer Wisconsin data set

Algorithm	Partition Coefficient index	Partition Entropy index	Fukuyama-Sugeno index	Xie-Beni index	Purity index	Normalized Mutual Information index
FCM	0,70180	0,65043	9,97471	0,17636	0,78141	0,20863
GK	0,80869	0,41571	3,11619	136,80074	0,77748	0,19883
GG	0,71908	0,61341	8,52217	30,85465	0,78417	0,20820
MID	0,91949	0,17782	8,03281	243,40205	0,76021	0,19926
FEBFC	0,83261	0,36418	3,60073	0,09625	0,75014	0,19384
FIDBFC	0,78031	0,47621	6,69028	0,10761	0,76131	0,20961

As we can see in Table 4.6, the MID algorithm gives the most accurate clustering results according to the Partition Coefficient and the Partition Entropy indices. According to the Fukuyama-Sugeno, Xie-Beni and Purity indices, clustering results of the GK, FEBFC and GG algorithms are the most accurate correspondingly.

The FIDBFC algorithm is the most accurate according to the Normalized Mutual Information index. It also more accurate than the original FEBFC algorithm according to the Purity index, but less accurate according to the other indices. Thus, for the Breast Cancer Wisconsin data set the FEBFC algorithm in general gives better results than its modification (the FIDBFC algorithm).

4.2.4. Indian Liver Patient Records

The Indian Liver Patient Records data set is located in the Kaggle machine learning repository [35]. The data set contains patient records consisting of some measurements and information whether a patient has liver disease. The records were collected from North East of Andhra Pradesh, India and can be used to construct a liver disease prediction system.

The data set contains *583 patterns*, represented by 11 attributes (*10 input attributes* and *1 output*), which are described in Table 4.7

Table 4.7. Attributes of the Liver Patient Records data set: **A_1-A_{10}** are input attributes and **C** is an output attribute

Attribute	Type	Description	Possible nominal values	Min. value	Max. value	Mean value	Standard deviation
A_1	numeric	Age of the patient	—	4	90	44.746	16.176
A_2	nominal	Gender of the patient	*Female*; *Male*	—	—	—	—
A_3	numeric	Total Bilirubin (mg/dL)	—	0.4	75	3.299	6.204
A_4	numeric	Direct Bilirubin (mg/dL)	—	0.1	19.7	1.486	2.806
A_5	numeric	Alkaline Phosphotase (IU/L)	—	63	2110	290.576	242.730
A_6	numeric	Alamine Aminotransferase (IU/L)	—	10	2000	80.714	182.464
A_7	numeric	Aspartate Aminotransferase (IU/L)	—	10	4929	109.911	288.671
A_8	numeric	Total Protiens (g/dL)	—	2.7	9.6	6.483	1.085
A_9	numeric	Albumin (g/dL)	—	0.9	5.5	3.142	0.795
A_{10}	numeric	Albumin and Globulin Ratio	—	0.3	2.8	0.947	0.318
C	nominal	Dataset	*1* – patient with liver disease; *2* – no disease	—	—	—	—

The fuzzy clustering algorithms described in Chapter 2 were performed on each numeric attribute of the Liver Patient Records data set. As a result, the membership functions, shown in Figures 4.19-4.24 were obtained.

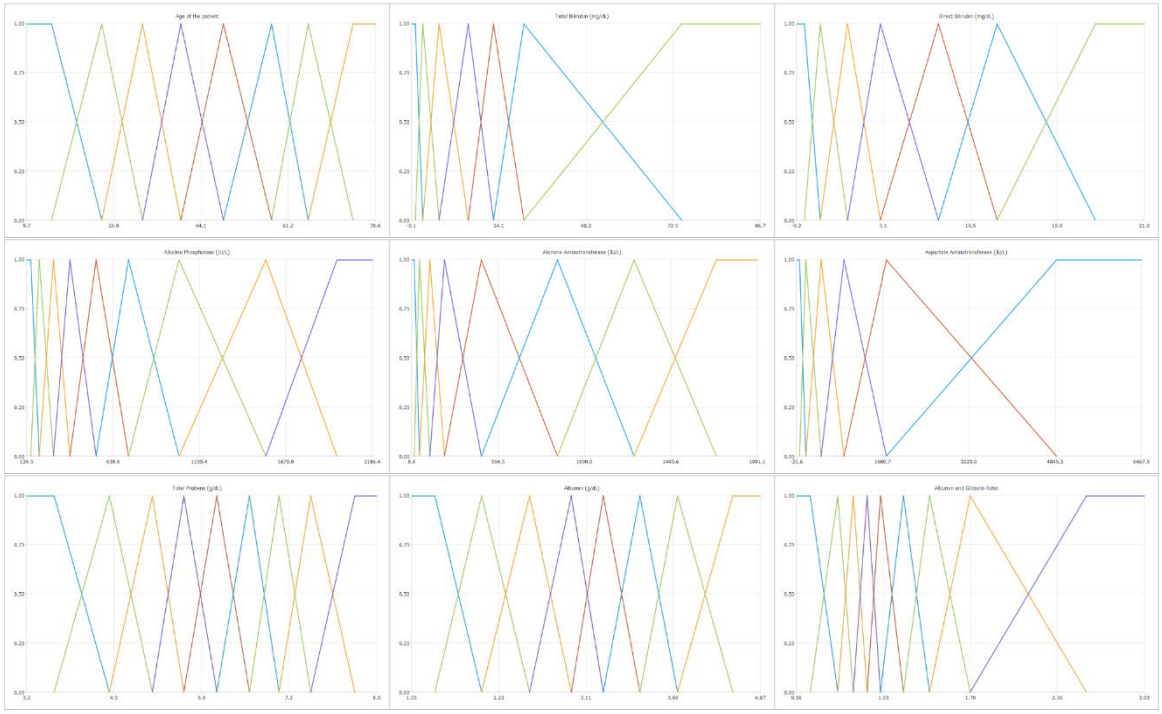


Figure 4.19. Membership functions of attributes $\mathbf{A_1}$, $\mathbf{A_3}$ – $\mathbf{A_{10}}$ obtained using the FCM algorithm with Pairing Frequency index for getting the optimal number of clusters

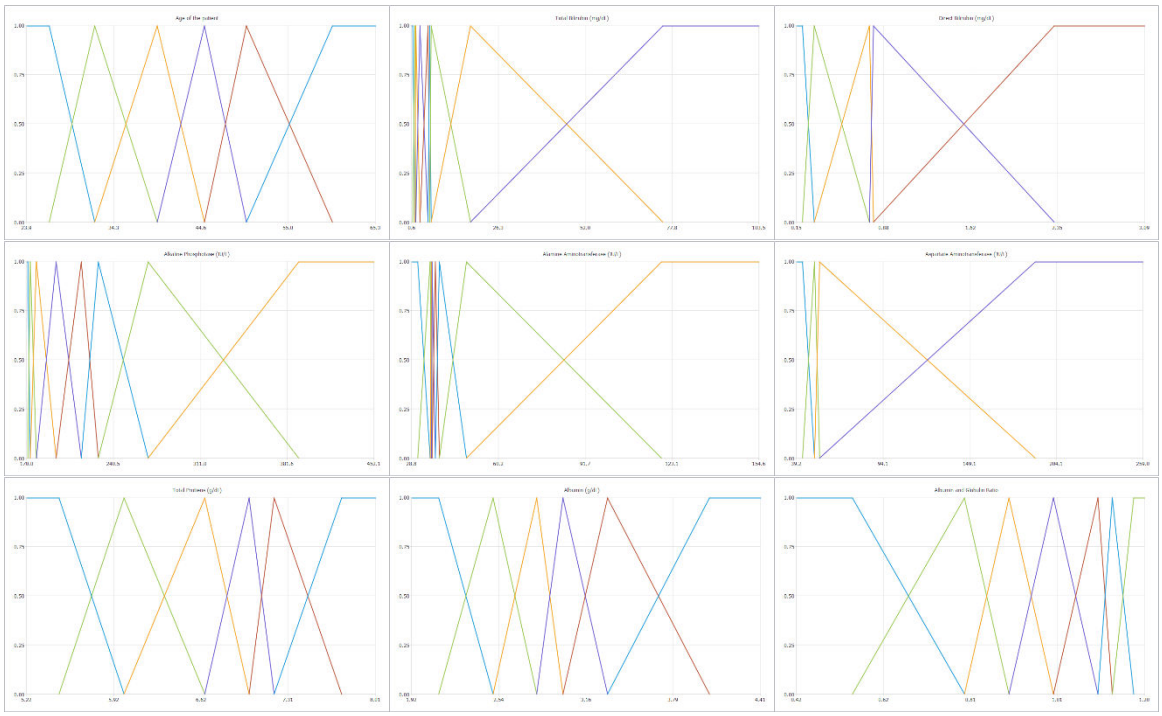


Figure 4.20. Membership functions of attributes $\mathbf{A_1}$, $\mathbf{A_3}$ – $\mathbf{A_{10}}$ obtained using the GK algorithm with Pairing Frequency index for getting the optimal number of clusters

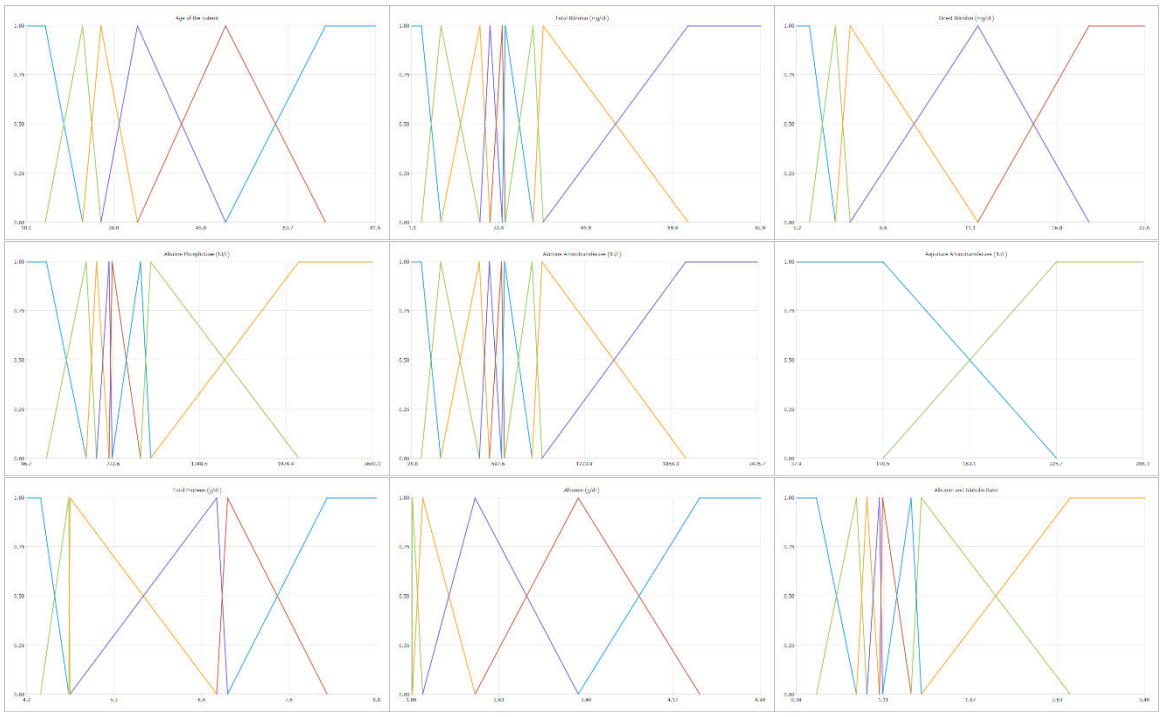


Figure 4.21. Membership functions of attributes **A1**, **A3** – **A10** obtained using the GK algorithm with Pairing Frequency index for getting the optimal number of clusters

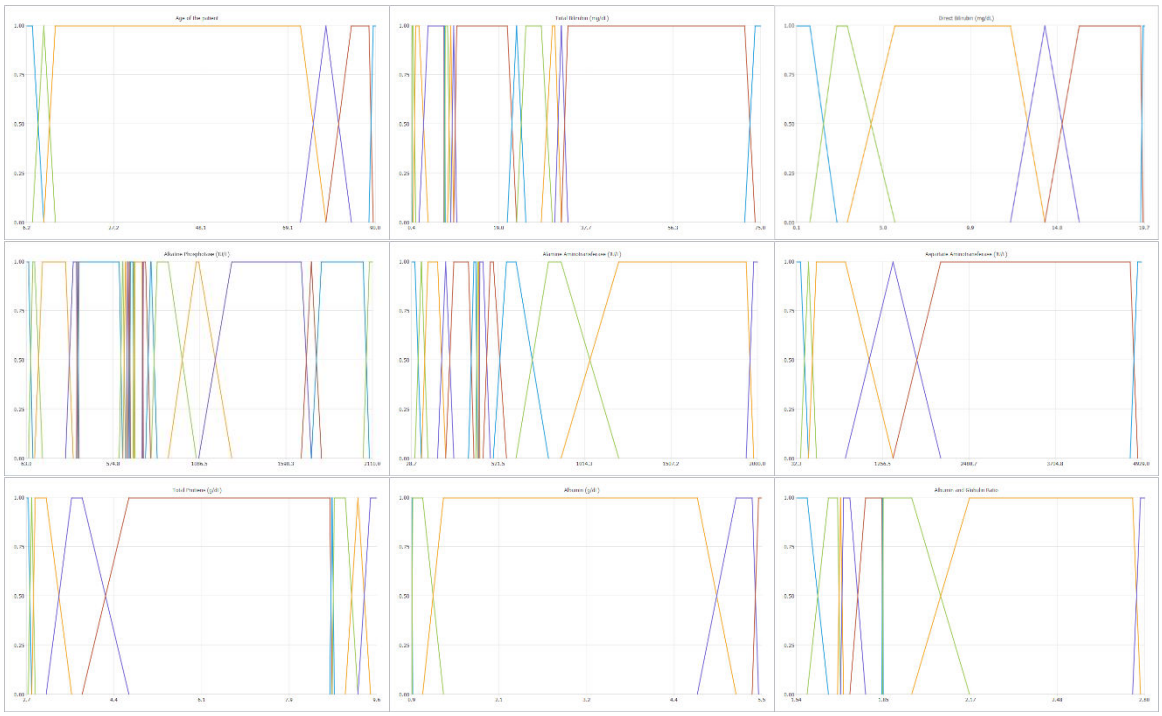


Figure 4.22. Membership functions of attributes **A1**, **A3** – **A10** obtained using the MID algorithm

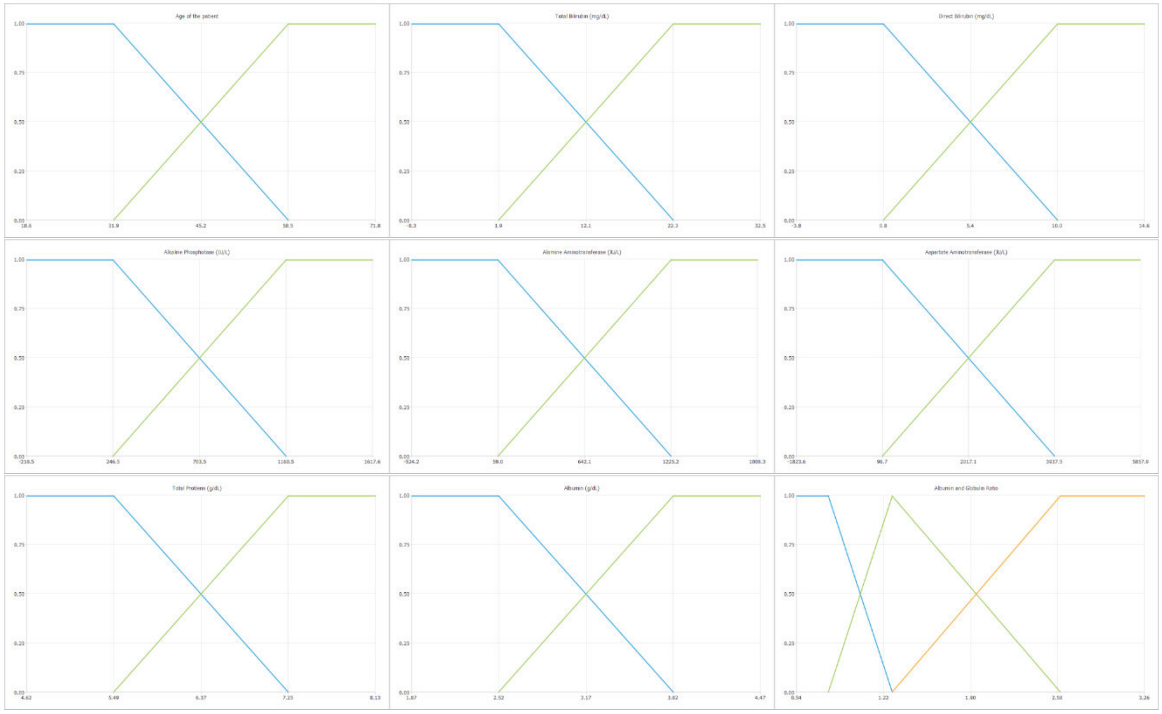


Figure 4.23. Membership functions of attributes $\mathbf{A_1}$, $\mathbf{A_3}$ – $\mathbf{A_{10}}$ obtained using the FEBFC algorithm

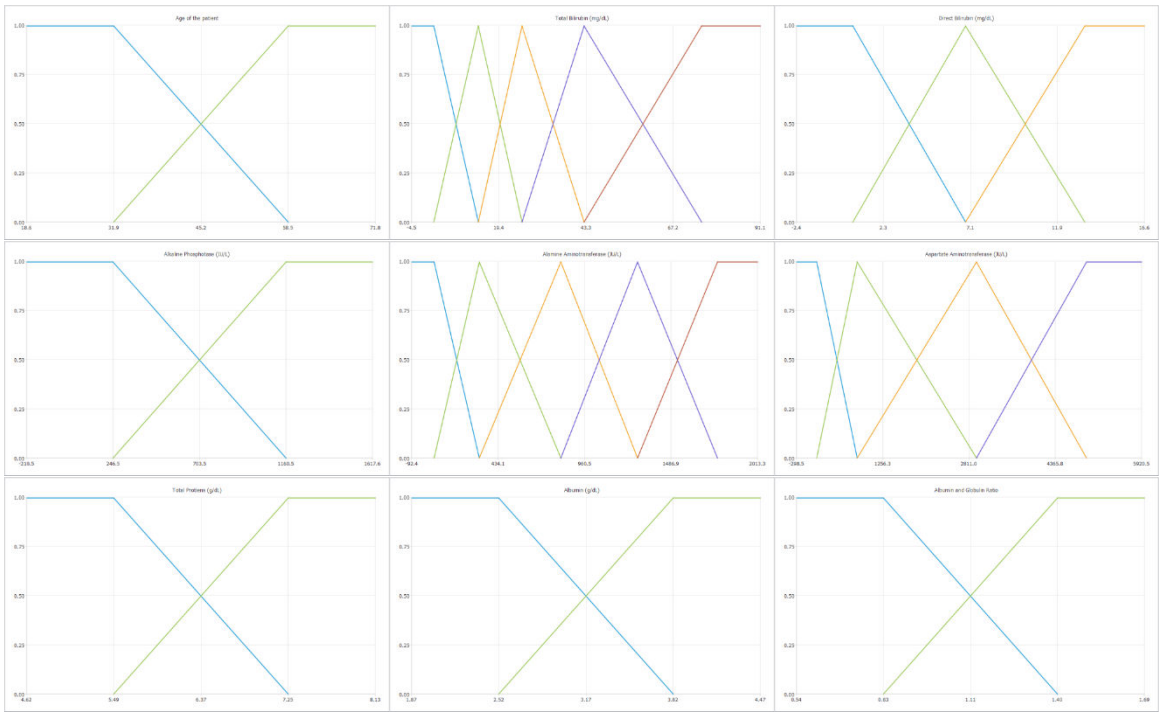


Figure 4.24. Membership functions of attributes $\mathbf{A_1}$, $\mathbf{A_3}$ – $\mathbf{A_{10}}$ obtained using the FIDBFC algorithm

Based on the obtained membership functions, fuzzification of the data set was made and the appropriate evaluation indices, described in the Section 4.1, were calculated. The results of calculation are shown in Table 4.8.

Table 4.8. Clustering Accuracy Indices calculated for the fuzzification performed on the Indian Liver Patient Records data set

Algorithm	Partition Coefficient index	Partition Entropy index	Fukuyama-Sugeno index	Xie-Beni index	Purity index	Normalized Mutual Information index
FCM	0,76521	0,51781	7,31219	0,23369	0,71355	0,03833
GK	0,84262	0,33978	4,81232	235,85757	0,71355	0,03798
GG	0,80658	0,41956	6,89524	5,91104	0,71355	0,03553
MID	0,94466	0,12144	7,53296	150,68521	0,71520	0,04287
FEBFC	0,88244	0,25931	3,02895	0,06235	0,71355	0,02400
FIDBFC	0,87023	0,28597	3,33581	0,06514	0,71355	0,02874

As we can see in Table 4.8, the MID algorithm gives the most accurate clustering results according to the Partition Coefficient, the Partition Entropy, the Purity and the Normalized Mutual Information indices. According to the Fukuyama-Sugeno as well as the Xie-Beni indices, clustering results of the FEBFC algorithm are the most accurate.

The FIDBFC algorithm is more accurate than the original FEBFC algorithm according to the Normalized Mutual Information index, but less accurate or has the same accuracy according to the other indices. Thus, for the Indian Liver Patient Records data set the FEBFC algorithm in general gives better results than its modification (the FIDBFC algorithm).

4.2.5. Chronic Kidney Disease

The Chronic Kidney Disease data set is located in the UCI machine learning repository [36]. This data set contains information whether a patient has chronic kidney disease and some additional measurements, that can be used to predict the disease.

The data set contains *400 patterns*, represented by 25 attributes (*24 input attributes* and *1 output*), which are described in Table 4.9

Table 4.9. Attributes of the Chronic Kidney Disease data set: **A_1-A_{24}** are input attributes and **C** is an output attribute

Attribute	Type	Description	Possible nominal values	Min. value	Max. value	Mean value	Standard deviation
A_1	numeric	Age (years)	—	2	90	51.483	16.954
A_2	numeric	Blood Pressure (mm/Hg)	—	50	180	76.469	13.459
A_3	nominal	Specific Gravity	*1.005*; *1.010*; *1.015*; *1.020*; *1.025*	—	—	—	—
A_4	nominal	Albumin	*0*; *1*; *2*; *3*; *4*; *5*	—	—	—	—
A_5	nominal	Sugar	*0*; *1*; *2*; *3*; *4*; *5*	—	—	—	—
A_6	nominal	Red Blood Cells	*normal*; *abnormal*	—	—	—	—
A_7	nominal	Pus Cell	*normal*; *abnormal*	—	—	—	—
A_8	nominal	Pus Cell clumps	*present*; *notpresent*	—	—	—	—
A_9	nominal	Bacteria	*present*; *notpresent*	—	—	—	—
A_{10}	numeric	Blood Glucose Random (mgs/dl)	—	22	490	148.037	74.689
A_{11}	numeric	Blood Urea (mgs/dl)	—	1.5	391	57.426	49.224
A_{12}	numeric	Serum Creatinine (mgs/dl)	—	0.4	76	3.072	5.610
A_{13}	numeric	Sodium (mEq/L)	—	4.5	163	137.529	9.193
A_{14}	numeric	Potassium (mEq/L)	—	2.5	47	4.627	2.816
A_{15}	numeric	Hemoglobin (gms)	—	3.1	17.8	12.526	2.713
A_{16}	numeric	Packed Cell Volume	—	9	54	38.885	8.141
A_{17}	numeric	White Blood Cell Count (cells/cumm)	—	2200	26400	8406.12	2520.06
A_{18}	numeric	Red Blood Cell Count (millions/cmm)	—	2.1	8	4.707	0.839
A_{19}	nominal	Hypertension	*yes*; *no*	—	—	—	—
A_{20}	nominal	Diabetes Mellitus	*yes*; *no*	—	—	—	—
A_{21}	nominal	Coronary Artery Disease	*yes*; *no*	—	—	—	—
A_{22}	nominal	Appetite	*good*; *poor*	—	—	—	—
A_{23}	nominal	Pedal Edema	*yes*; *no*	—	—	—	—
A_{24}	nominal	Anemia	*yes*; *no*	—	—	—	—
C	nominal	Class	*ckd*; *notckd*	—	—	—	—

The fuzzy clustering algorithms described in Chapter 2 were performed on each numeric attribute of the Chronic Kidney Disease data set. As a result, the membership functions, shown in Figures 4.25-4.30 were obtained.

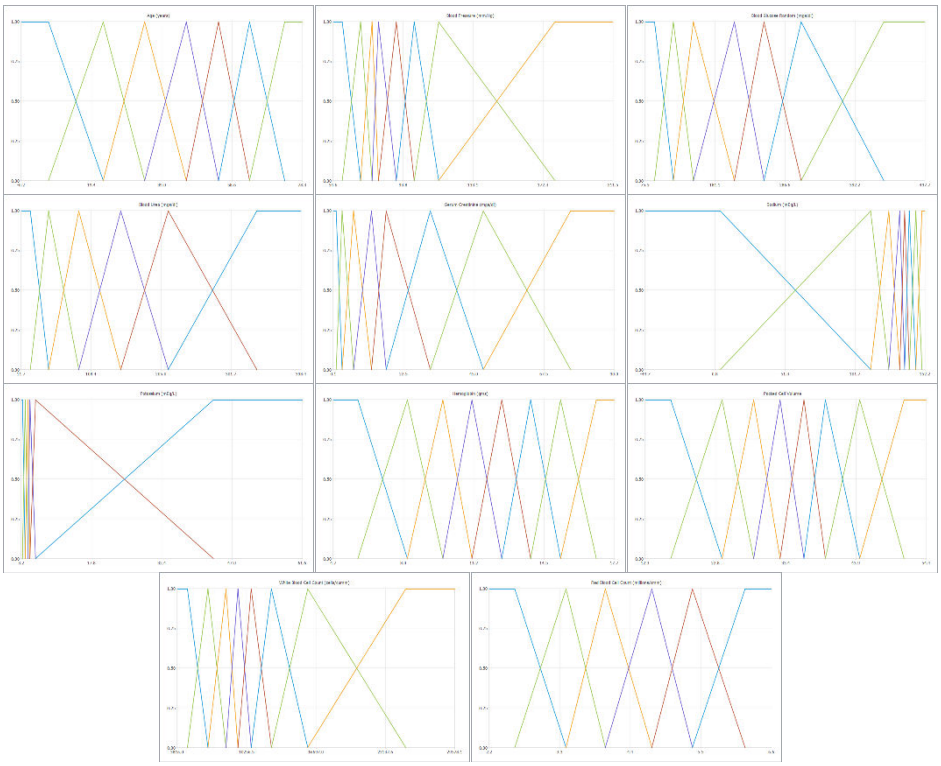


Figure 4.25. Membership functions of attributes **A1**, **A2**, **A10** – **A18** obtained using the FCM algorithm with Pairing Frequency index for getting the optimal number of clusters

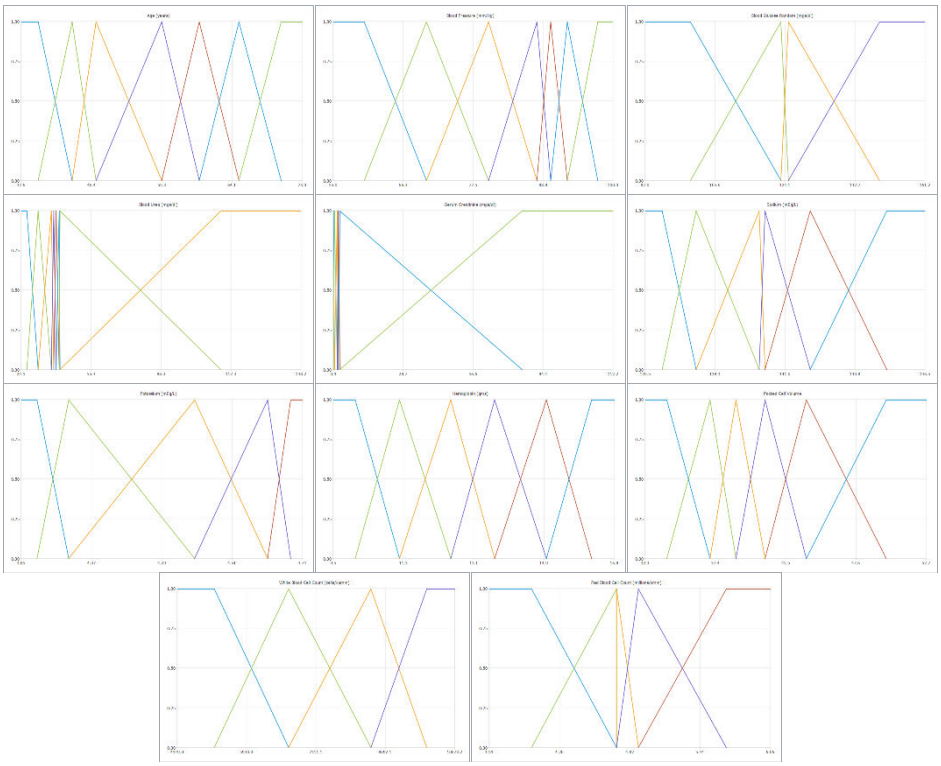


Figure 4.26. Membership functions of attributes **A1**, **A2**, **A10** – **A18** obtained using the GK algorithm with Pairing Frequency index for getting the optimal number of clusters

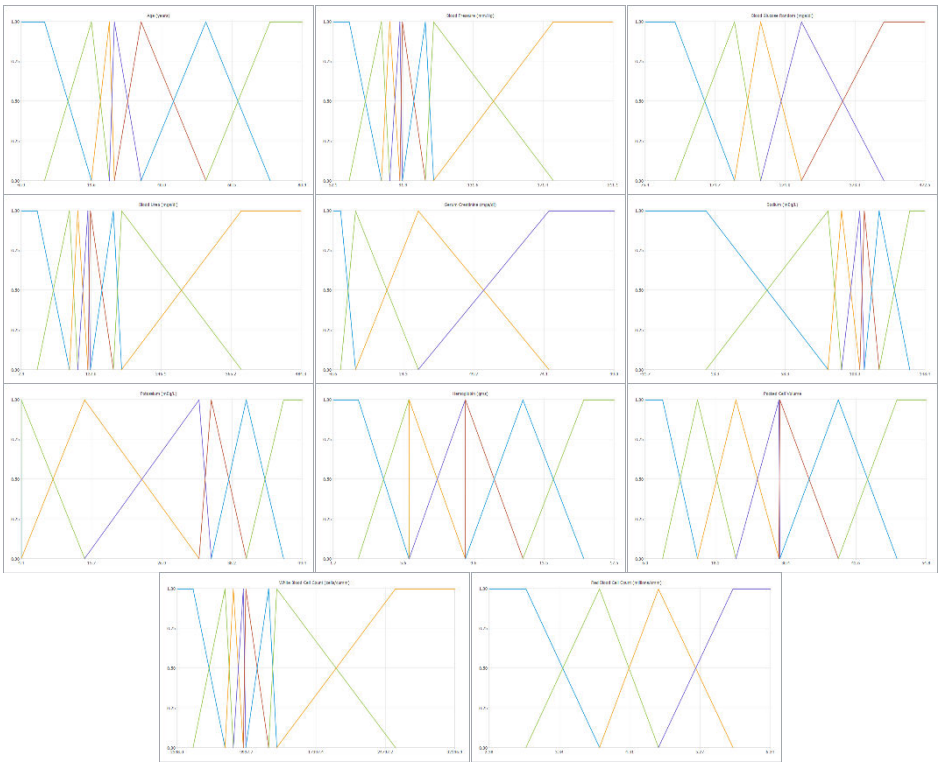


Figure 4.27. Membership functions of attributes **A1**, **A2**, **A10** – **A18** obtained using the GG algorithm with Pairing Frequency index for getting the optimal number of clusters

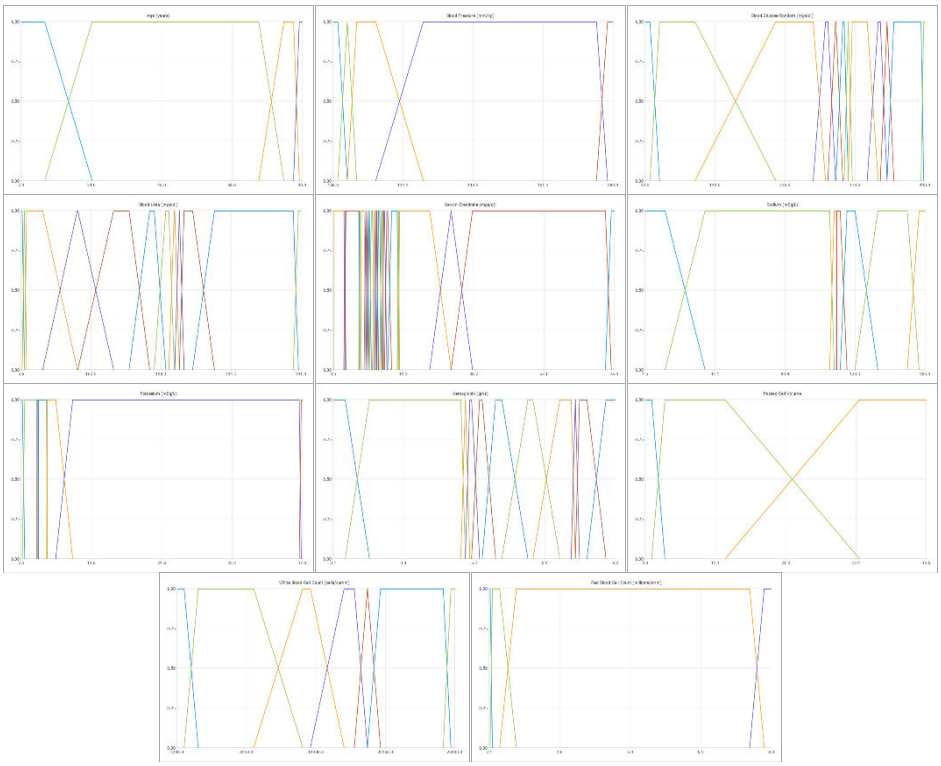


Figure 4.28. Membership functions of attributes **A1**, **A2**, **A10** – **A18** obtained using the MID algorithm

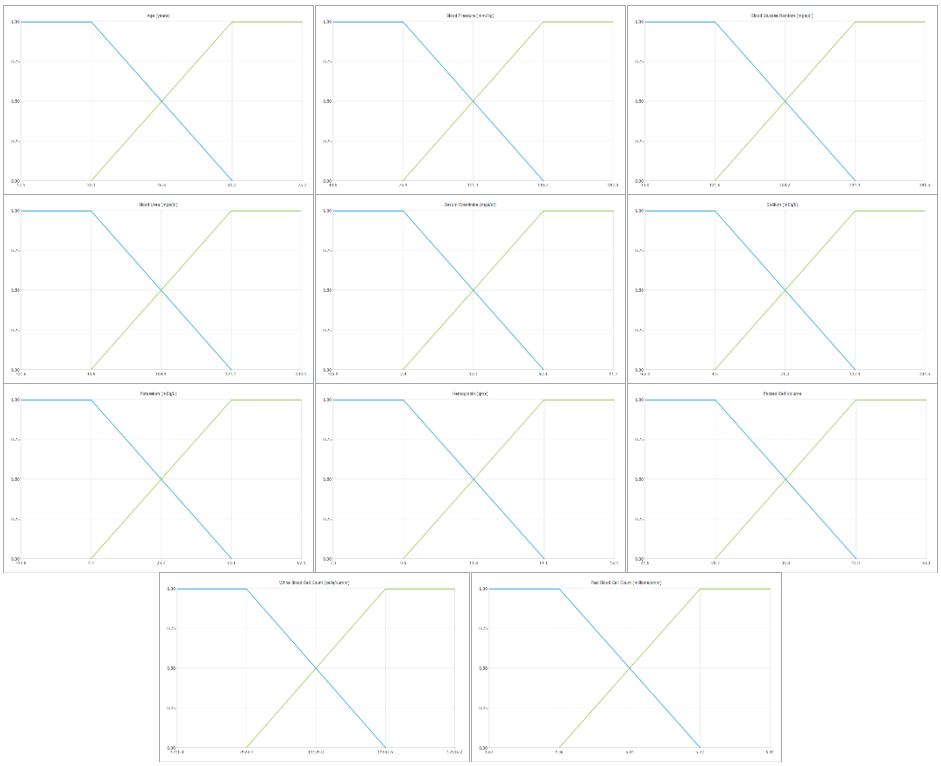


Figure 4.29. Membership functions of attributes **A1**, **A2**, **A10** – **A18** obtained using the FEBFC algorithm

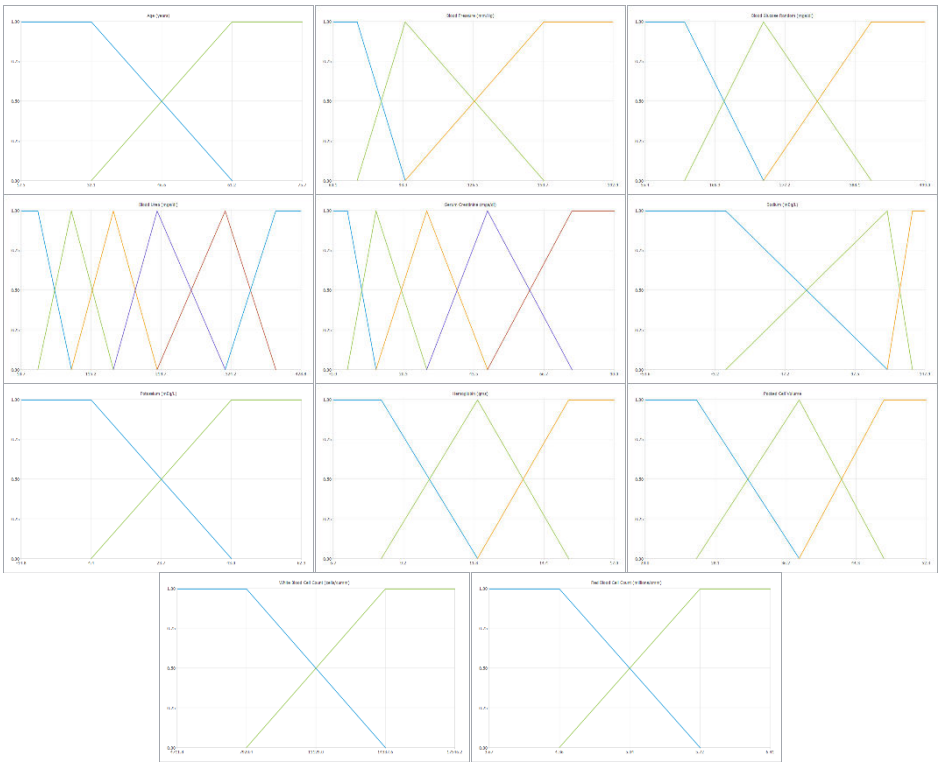


Figure 4.30. Membership functions of attributes **A1**, **A2**, **A10** – **A18** obtained using the FIDBFC algorithm

Based on the obtained membership functions, fuzzification of the data set was made and the appropriate evaluation indices, described in the Section 4.1, were calculated. The results of calculation are shown in Table 4.10.

Table 4.10. Clustering Accuracy Indices calculated for the fuzzification performed on the Chronic Kidney Disease data set

Algorithm	Partition Coefficient index	Partition Entropy index	Fukuyama-Sugeno index	Xie-Beni index	Purity index	Normalized Mutual Information index
FCM	0,77893	0,49223	5,33962	0,22215	0,74692	0,17449
GK	0,86890	0,28460	2,22786	169,93521	0,75999	0,19142
GG	0,79741	0,44822	4,80674	22,14820	0,68667	0,13921
MID	0,93745	0,13790	5,55817	301,79223	0,63449	0,07141
FEBFC	0,87330	0,28549	1,50738	0,06734	0,65568	0,09795
FIDBFC	0,82725	0,38268	3,01684	0,08254	0,67831	0,12089

As we can see in Table 4.10, the MID algorithm gives the most accurate clustering results according to the Partition Coefficient and the Partition Entropy indices; the FEBFC algorithm gives the most accurate clustering results according to the Fukuyama-Sugeno and the Xie-Beni indices; the GK algorithm gives the most accurate clustering results according to the Purity and the Normalized Mutual Information indices.

The FIDBFC algorithm is more accurate than the original FEBFC algorithm according to the Purity and the Normalized Mutual Information indices, but less accurate according to the other indices. Thus, for the Chronic Kidney Disease data set the FEBFC algorithm in general gives better results than its modification (the FIDBFC algorithm).

CONCLUSION

The diploma thesis was devoted to problematics of transformation from numeric into linguistic values, that is based on clustering, in medical data analysis. A study of theoretical aspects of cluster analysis was performed for understanding the problematics. In a process of the study several definitions of the term "clustering" were considered and own definition was proposed by the author of the thesis, according to which, clustering is an unsupervised learning method of partitioning a data set into a finite number of discrete groups (clusters) such that each group consists of similar objects according to some defined distance measure. The most commonly used data types in clustering are discrete and continuous types. The discrete data type aggregates nominal and binary (symmetrical as well as asymmetrical) types. The study also has involved describing of similarity and dissimilarity measures and explanation differences between hard clustering and fuzzy clustering.

The software tool, that is be able to make transformation of values of any numeric attribute of a medical data set into fuzzy values, has been developed to achieve the goal of this thesis. The main requirement for the tool was the ability of integration into an expert system for medical data analysis, in a way that it could be considered as a computational component of the system. This goal was achieved successfully by implementation of fully-featured Fuzzy Clustering Tool. This tool has two basic functionalities: performing a cluster analysis to transform numeric into linguistic values and fuzzification based on the calculated membership functions to obtain a resulting fuzzy data set. In addition to these basic functionalities, the Fuzzy Clustering Tool provides simple statistical analysis of a data set, visualization of a membership function and a possibility to import external clustering results.

The following clustering algorithms were implemented in the Fuzzy Clustering Tool to perform a clustering analysis: Fuzzy c-Means, Gustafson-Kessel algorithm, Gath-Geva algorithm, Multi-Interval Discretization and Fuzzy Entropy Based Fuzzy Classifier. Additionally, the author of the thesis proposed a modification of the Fuzzy Entropy Based Fuzzy Classifier, that supposes using a fuzzy information density instead of fuzzy entropy. He called this modification Fuzzy Information Density Based Fuzzy Classifier (FIDBFC). The FIDBFC was formally described in Section 2.5 and its implementation was included into the Fuzzy Clustering Tool.

All implemented clustering algorithms were evaluated and compared on the following medical data sets: Pima Indians Diabetes, Heart Disease, Breast Cancer Wisconsin, Indian Liver Patient Records and Chronic Kidney Disease. In the first two of them the proposed modification of the clustering algorithm (i.e. FIDBFC) gave more accurate clustering results than the original FEBFC algorithm. Thus, we can state that on some medical data set using the proposed FIDBFC for transformation from numeric into linguistic and fuzzy values is preferable.

BIBLIOGRAPHY

1. Zaitseva E., Levashenko V., Kvassay M., Barach P. *Healthcare system reliability analysis addressing uncertain and ambiguous data.* 2017 International Conference on Information and Digital Technologies (IDT), Žilina, 2017, pp. 442-451.
2. Barach P., Levashenko V., Zaitseva E. *New Methods for Healthcare System Evaluation Using Human Reliability Analysis.* Proceedings of the Human Factors and Ergonomics Society, vol. 61(1), 2017, pp. 583-587.
3. Rokach Lior, Maimon Oded. *Data Mining with Decision Trees. Theory and Applications* (Series in Machine Perception and Artificial Intelligence: Volume 69). World Scientific Publishing Co. Pte. Ltd., 2008, 263 pages.
4. Gan G., Chaoqun M., Jianhong W. *Data Clustering: Theory, Algorithms, and Applications.* ASA-SIAM Series on Statistics and Applied Probability, SIAM, Philadelphia, ASA, Alexandria, VA, 2007, 488 pages.
5. Abonyi J., Feil B. *Cluster Analysis for Data Mining and System Identification.* Birkhäuser Verlag AG, 2007, 319 pages.
6. Xu Rui, Wunsch Donald C. II. *Clustering*. John Wiley & Sons, 2009, 358 pages.
7. Zhang G. P. *Neural Networks for Data Mining.* In: Maimon O., Rokach L. (eds) Data Mining and Knowledge Discovery Handbook. Springer, Boston, MA, 2009, pp. 17-44.
8. Pirim H., Eksioglu B., Perkins A., Yuceer C. *Clustering of high throughput gene expression data.* Computers & Operations Research vol. 39, 2012, pp. 3046–3061.
9. Hopper F., Klawonn F., Kruse R., and Runkler T. *Fuzzy Cluster Analysis: Methods for Classification, Data Analysis, and Image Recognition.* John Wiley & Sons, 1999, 289 pages.
10. Pedrycz Witold. *Knowledge-based clustering: from data to information granules.* John Wiley & Sons, 2005, 316 pages.
11. Korolyuk I. P. *Medical informatics* [transl. from orig. rus. *Медицинская информатика*]. Samara : Ofort Ltd., 2012, 244 pages.
12. Albayrak S., Armasyali F. *Fuzzy C-Means Clustering on Medical Diagnostic Systems.* In International XII Turkish Symposium on Artificial Intelligence and Neural Networks, 2003.

13. Chen H., Fuller S. S., Friedman C., Hersh W. *Knowledge Management, Data Mining, and Text Mining in Medical Informatics*. In: Chen H., Fuller S. S., Friedman C., Hersh W. (eds) Medical Informatics. Integrated Series in Information Systems, vol 8. Springer, Boston, MA, 2005, pp. 3-33.
14. Liao M., Li Y., Kianifard F., Obi E., Arcona S. *Cluster analysis and its application to healthcare claims data: a study of end-stage renal disease patients who initiated hemodialysis.* BMC Nephrol. 2016, 17(4): 25.
15. Babuška Robert. *Fuzzy Clustering Algorithms with Applications to Rule Extraction.* In: Szczepaniak P.S., Lisboa P.J.G., Kacprzyk J. (eds) Fuzzy Systems in Medicine. Studies in Fuzziness and Soft Computing, vol 41. Physica, Heidelberg, 2000, pp. 139-173.
16. Bezdek J. C. *Cluster validity with fuzzy sets.* Journal of Cybernetics, vol. 3(3), 1973, pp. 58-72.
17. Bezdek J. C. *Mathematical models for systematics and taxonomy.* In: Proceedings 8th International Conference in Numerical Taxonomy, Freeman, San Francisco, 1975, pp. 143-166.
18. Fukuyama Y., Sugeno M. *A new method of choosing the number of clusters for the fuzzy C-means method.* In Proc. 5th Fuzzy Syst. Symp., 1989, pp. 247-250.
19. Xie X. L., Beni G. *A validity measure for fuzzy clustering.* IEEE Trans. Pattern Anal. Mach. Intell., vol. 13, no. 8, Aug. 1991, pp. 841–847.
20. Zhou K., Ding S., Fu C., Yang S. *Comparison and weighted summation type of fuzzy cluster validity indices.* Int J Comput Commun Control, vol. 9, 2014, pp. 370-378.
21. Pakhira M. K., Bandyopadhyay S., Maulik U. *Validity Index for Crisp and Fuzzy Clusters.* In. Pattern Recognition, vol. 37(3), 2004, pp. 487-501.
22. Bezdek J. C. *Pattern recognition with fuzzy objective function algorithms.* New York, NY: Plenum Press, 1981, 272 pages.
23. Dunn J. *A fuzzy relative of the ISODATA process and its use in detecting compact well separated clusters.* Journal of Cybernetics, 3 (3), 1974, pp. 32-57.
24. Gustafson D. E., Kessel W. C. *Fuzzy clustering with a fuzzy covariance matrix.* Proc. IEEE CDC, 1979, pp. 761-766.
25. Gath I., Geva A. B. *Unsupervised optimal fuzzy clustering.* IEEE Trans. Patt. Anal. Machine Intell., vol. 11, no 7, 1989, pp. 773-780.

26. Cui H., Zhang K., Fang Y., Sobolevsky S., Ratti C., Horn B. K. P. *A Clustering Validity Index Based on Pairing Frequency.* In IEEE Access, vol. 5, 2017, pp. 24884-24894.
27. Popel D. V. *From continuous to multiple-valued data.* In Proceedings IEEE International Symposium on Multiple-Valued Logic, 2003, pp. 367-372.
28. Levashenko V., Zaitseva E., Kovalík Š., *Projektovanie systémov pre podporu rozhodovania na základe neurčitých dát.* Žilina : Žilinská univerzita, 2013, 245 pages.
29. Lee H. M., Chen Ch. M., Chen J. M., Jou Y. L. *An efficient fuzzy classifier with feature selection based on fuzzy entropy.* In IEEE Transactions on systems, Man, and Cybernetics – Part B: Cybernetics, vol. 31, no. 3, 2001, pp. 426-432.
30. Kosko B. *Fuzzy entropy and conditioning.* Information sciences, vol. 40, 1986, pp. 165-174.
31. Rendón E., Abundez I., Arizmendi A., Quiroz E. M. *Internal versus external cluster validation indexes.* International Journal of Computers and Communications, vol. 5(1), 2011, pp. 27-34.
32. Rezaee M. R., Lelieveldt B. P. F., Reiber J. H. C. *A new cluster validity index for the fuzzy c-mean.* Pattern Recognition Letters, vol. 19, 1998, pp. 237–246.
33. Sripada S. Ch., Rao M. S. *Comparison of purity and entropy of K-means Clustering and Fuzzy C means Clustering.* Indian Journal of Computer Science and Engineering (IJCSE), Vol. 2, No. 3, 2011, pp. 343-346.
34. Kvålseth T. O. *On Normalized Mutual Information: Measure Derivations and Properties.* Entropy, vol. 19(11), 2017, 631 pages.
35. *The Home of Data Science & Machine Learning*, https://www.kaggle.com.
36. *UCI Machine Learning Repository: Center for Machine Learning and Intelligent Systems*, https://archive.ics.uci.edu/ml/index.php.

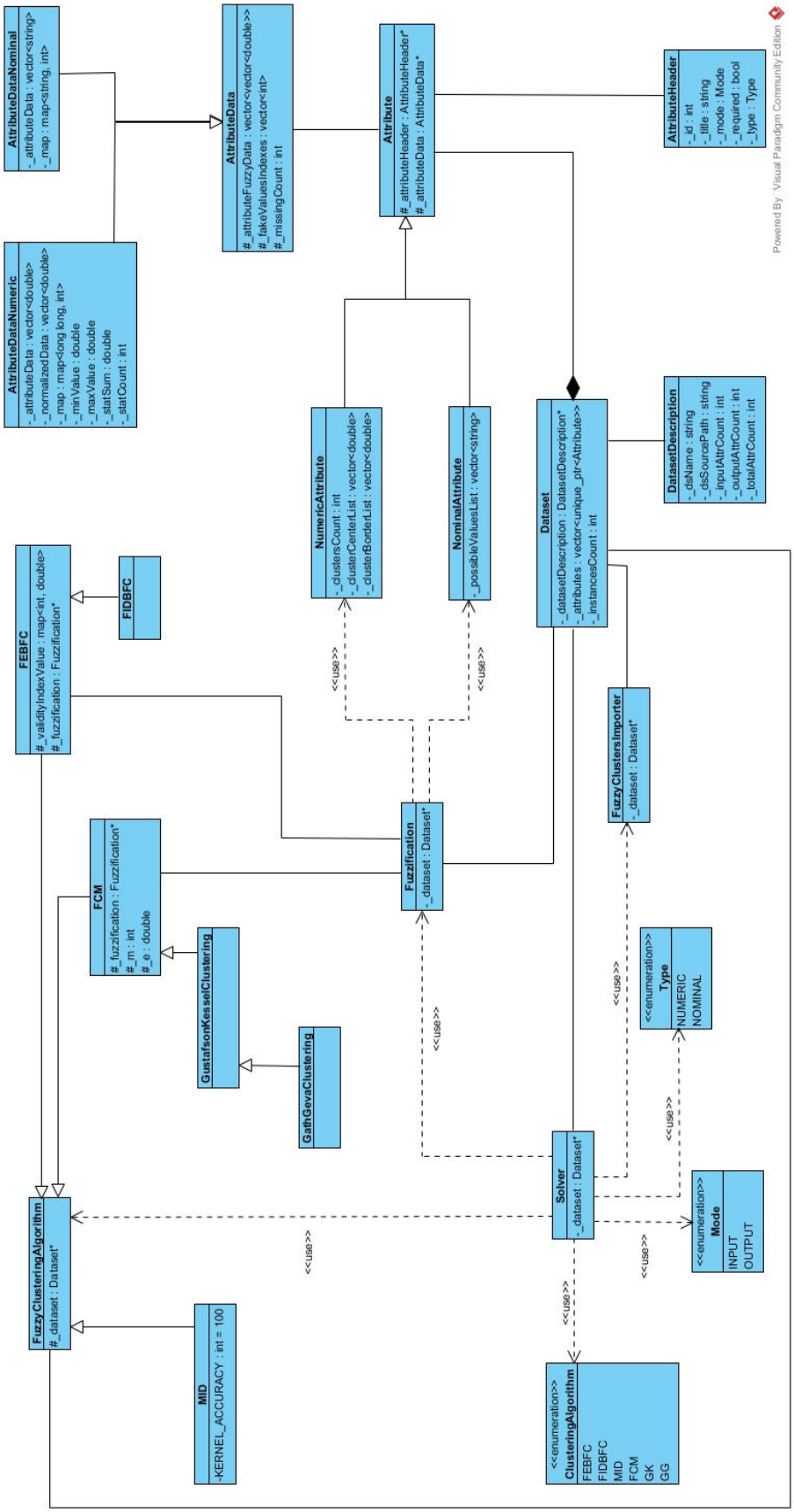
List of Appendices

Appendix 1 UML Class diagram of the Fuzzy Clustering Tool

Appendix 2 Contents of the CD

Appendices

Appendix 1: UML Class diagram of the Fuzzy Clustering Tool



Appendix 2: Contents of the CD

The enclosed CD contains:

- the diploma thesis in electronic form (PDF format);
- the source code of the Fuzzy Clustering Tool.