POLITECHNIKA WROCŁAWSKA
WYDZIAŁ ELEKTRONIKI

KIERUNEK: INFORMATYKA (ANG.)

PRACA MAGISTERSKA

Hybrydowy klasyfikator wykorzystujący zbiory rozmyte i przbliżone-badania symulacyjne

Hybrid classifier using fuzzy logic and rough sets- simulation investigations

AUTOR:
Marcin Majak

OPIEKUN:
dr inż. Andrzej Żołnierek, W4/K2

OCENA PRACY:

WROCŁAW 2010

Streszczenie

WROCLAW UNIVERSITY OF TECHNOLOGY
FACULTY OF ELECTRONICS

AREA: ADVANCED INFORMATICS AND CONTROL (AIC)

MASTER THESIS

Hybrid classifier using fuzzy logic and rough sets- simulation investigations

Hybrydowy klasyfikator wykorzystujący zbiory rozmyte i przbliżone-badania symulacyjne

AUTHOR:
Marcin Majak

SUPERVISOR:
Andrzej Żołnierek, PhD, W4/K2

GRADE:

WROCŁAW 2010

Abstract

Contents

1 Introduction

1.1 Preface

With the improvement of computers we are able to tackle with high dimensional problems in control or pattern recognition task. At first glance everything looks so easy, but when we go deeper into a problem more and more problems become evident and unavoidable. For example pattern can be described by many attributes. Some of them are very meaningful while others brings only noise and distortion. This is the role of the algorithm to pick valuable attributes, but finding a minimal attribute reduct is $\mathcal{NP}$-hard problem. Another obstacle is connected with overlaps in the attribute space. When attributes are easily separable even a simple classifier works perfectly, but for more tricky cases very sophisticated approaches have to be applied.

For the past years, many scientist in the world tried to invent new algorithms to improve the accuracy of classification or optimize control of the plant. There are many solutions, but ones of the most eminent in the literature are: neural networks, fuzzy logic or evolutionary algorithms such as genetic algorithm. Because simple approaches failed in more complicated problem, scientist tried to applied algorithms for dimensionality reduction and merge abilities of single classifier into combined one. This improved the quality of classification significantly, but until then no one has managed to invent such a classifier that will never make no mistakes.

This thesis touches the broad topic of pattern recognition task which is very difficult and demanding problem in every aspect of science. Generally, pattern classification is about assigning label to an unknown object based on the available knowledge coming from training dataset or expert experience. This process be compared to the capability of human brain which is able to put certain scenario into context and identify distinguishable object components. The whole process of classification can be broken down into few parts and each phase has a significant impact on the final results (more detailed description will be presented in section 2).

1.2 Main goals of the thesis

This thesis concerns the problem of pattern recognition. As it was previously said, in the literature one can find many algorithms used for object classification, while here the rough sets, fuzzy logic and genetic algorithms are used and investigated. The general purpose of this thesis is to propose a hybrid classifier using the power of fuzzy logic and rough sets.

First of all the mathematical background and basic properties of these algorithms will be presented and later simulation investigation will be carried out to prove the usefulness of proposed fusion. It should be noted that for the simulation purposes author implemented basic fuzzy logic, rough sets algorithms and created a hybrid classifier.

This thesis is a continuation of work in the field of pattern recognition. The results of previous experiments can be found in [36], [37]. The aim of this paper is to collect all experiments scenarios in one place and draw conclusions. It is meant to show the steps of how to construct an advanced hybrid classifier from the basic algorithms. At first, the basic properties of rough sets algorithm are checked in simulations, later the algorithm for modification decision rules is introduced and at the end as the final point the hybridization of algorithms is proposed using fuzzy logic, rough sets and genetic algorithm.

At the end of this section, the experiment environment should be characterized in few words. Given is the problem of pattern recognition (classification): in this thesis datasets from UCI Repository will be used (described in section 4). Those datasets are broadly available and everyone in the future would be able to repeat the simulation investigation and compare the results with those presented in this paper. To find is the algorithm classification accuracy on the given dataset. Figure 1.1 shows the general schematic of the simulation environment.

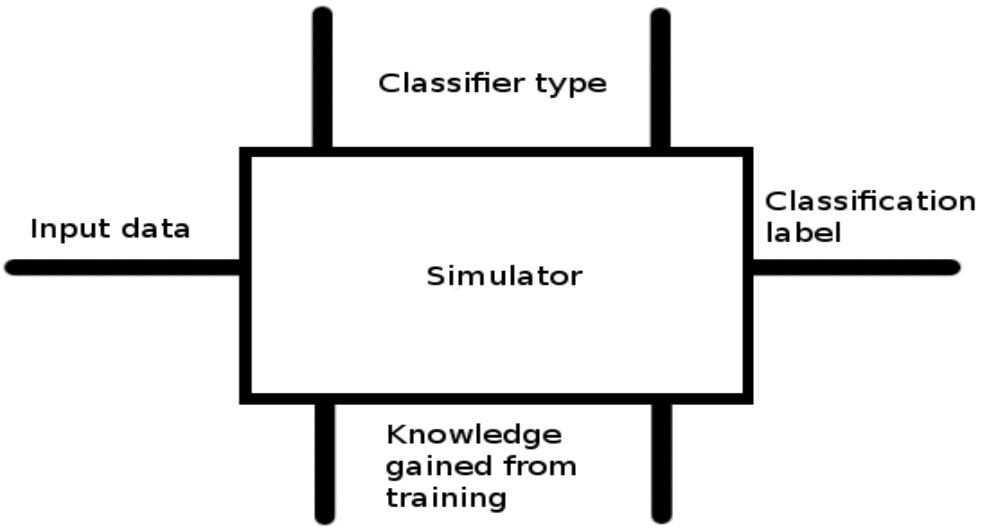

Figure 1.1: General schema of simulation environment. Input/output of the system

1.3 Scope of this project

This paper comprises of two parts. In the first, the review of literature and the basic notation used in the whole paper is presented. Sections 2 is the source of the basic knowledge about pattern recognition. Sections 2.3, 2.4 presents the description of algorithms used in this thesis. Additionally, this part shows the algorithm construction steps using pseudo-code.

The second part, which is the main point of this thesis, presents experiments analysis. Paragraph 4 describes the experiment environment setting and program written in *Python* to simulate all implemented algorithms. Results with comments are placed in section 5. General conclusions and plans for the future work are placed in sections 6, 7, respectively.

2 Pattern recognition algorithms

2.1 Introduction

This section is devoted for presenting information about pattern recognition algorithms. The main purpose is to describe algorithms that are used in this thesis.

Pattern recognition is a wide area of science in which we are interested in assigning label from a given set of classes to every unknown pattern. The whole process of classification can be divided into few phases, where each has a significant impact on the final classification accuracy:

1. Data collection
2. Feature selection
3. Model selection
4. Classifier selection
5. Training
6. Testing

At first in this section few basic information and categories will be given about pattern recognition. Generally, the whole process of classification can be broken down into two main categories:

- supervised - incoming to the system objects are not previously labeled and this is the system task to find an appropriate structure of the data, to establish the organization of the classes basing only on the available data, there is no statistical or expert knowledge at a hand.

- unsupervised - in this approach incoming patterns have labels and can be treated as a training set. This allows the classifier can retrieve information from data

In this thesis supervised learning is reconsidered because available datasets are labeled with class number. Further division is connected with syntactic and statistical approach. In former, each pattern is represented in terms of d features (measurements) and is viewed as a point in d-dimensional feature space, while the latter is based on the characterization of the inherent structure of the qualitative features. For that reason, the complex patterns can be decomposed using a hierarchical structure in simple sub-patterns. The patterns are viewed as sentences belonging to a language, primitives represents the alphabet of the language and the sentences are generated according to a grammar which is inferred from the available training data. EKG waveforms, textured images and shape analysis of contours are the examples of syntactic approach.

2.2 Problem statement for pattern recognition task

In this section the problem statement will be presented in case of pattern recognition algorithm. For the purpose of this thesis we assume a supervised learning and denote each pattern by the label $j \in M$, where M is an m-element set of possible states numbered with the successive natural numbers. The state j is unknown and does not undergo the direct investigation. What can only be measured are attributes or features by which a state manifests itself. For this reason each object(pattern) will be described by a d-dimensional measured feature vector $x \in X$. In order to classify unknown pattern algorthm uses knowledge stored in the training set consisting of N training patterns:

$$S = (x_1, j_1), (x_2, j_2), \ldots, (x_N, j_N) \tag{2.1}$$

In practice the decision algorithm with learning phase should use knowledge included in the training set S and as the consequence the algorithm with learning is of the following form:

$$i = \Psi(S, x),\ i \in M \tag{2.2}$$

In decision theory, to ensure that Ψ approximates the problem as closely as possible an additional loss function is introduced that assigns a specific value to the loss resulting from producing an incorrect label. The particular loss function depends on the type of label being predicted. In case of classification problem it is zero-one loss function. This corresponds simply to assigning a loss value of 1 to any incorrect labeling and is equivalent to computing the accuracy of classification procedure over the set of training data.

2.3 Rough sets

2.3.1 Introduction

Rough sets theory represents the mathematical approach to deal with imperfect knowledge. In the standard standard pattern recognition task we need precise information about pattern to recognize, while rough sets can deal with vague or incomplete data. The problem of imperfect information has been tackled for a long time and it has become a crucial issue for many scientist. One of the most prominent approaches in the recent years are fuzzy logic and rough sets. In this section the latter approach is presented in greater details.

Comparing with other methods, rough sets have many advantages, but one of the most important one is its ability to works only on the raw data, with no additional information such as density probability in Bayesian algorithm [40], [15]. The main facts about rough sets algorithm can be summarized in few point presented below:

1. simplifies data by the granulation pre-processing
2. is able to reduce attributes
3. generates set of easy to understand and readable decision *IF-THEN* rules
4. evaluates significance of data

When talking about rough set theory one has to understand the concept of a set and how a rough set is related to the classical set represented in mathematic. From the mathematical point of view the crisp (precise) set is a collection of objects of interest and is uniquely determined by its elements. In other words, it means that every element must be uniquely classified as belonging to the set or not (true or false). For example, the set of odd numbers is crisp because every number is either odd or even and cannot be partially in both.

Generally, we come across problem form the nature that are much more complicated than simple decision whether an objects belong to the set or not. For some sets we cannot precisely describe element membership. Reconsider the group of people and division into set of small and high people. The height is not a precise but a vague concept and data vagueness can be met in many problems found in the nature. Here is the spot for rough sets theory where vagueness is expressed by a boundary region of a set.

2.3.2 Basic notation

In rough sets theory to represent datasets (information) we introduce a notion called an *information system* [39], [41]. It can be described by 4-tuple

$$IS =< U, Q, V, f > \tag{2.3}$$

where the notation used in eq. (2.3) is as follows:

- U is the universe of discourse which is a finite set of objects (patterns)
- Q is a finite set of attribute by which each patterns manifests itself (is described)
- $V = \bigcup V_q$, V_q represents a domain of attribute q
- $f : U \times Q \to V$ is a total information function, such that $\bigvee_{q \in Q, x \in U} f(x, q) \in U$

The information system can be represented as a finite table in which columns are labeled by attributes and each rows stands for an object from IS. Over the information table we can define decision table T where the set of attributes Q is disjoined into two subset C and D. The set C is a subset of condition attributes, and the set D contains decision attributes by which we can partition set U into decision classes.

From the granular nature of rough sets it may happen that some objects in the U are indistinguishable due to the limited information caused by granulation process. Taking this into account let define an indiscernibility relation $R \to U \times U$, representing the lack of knowledge about patterns in the set U. The indiscernibility relation on U can be extended and associated with every non-empty subset of attributes $P \subseteq Q$ and is defined by eq. (2.4)

$$I_P = \{(x, y) \in U \times U : f(x, q) = f(y, q), \bigvee_{q \in P}\} \tag{2.4}$$

Now having I_P we can say that objects x and y are P-indiscernible by a set of attributes P if $y \, I_P \, x$. Relation I_P divides the set U into blocks (concepts) of P-indiscernible objects. The P-elementary set containing objects P-indiscernible with $x \in U$ is referred as $I_P(x)$ and defined as follows:

$$I_P = \{y \in U : y \, I_P \, x\} \tag{2.5}$$

By representing a target concept X as a subset of U we would like to describe it with respect to R. Additionally let assume P as non-empty subset of attributes from Q. In rough sets reasoning an object membership to a set can be represented in two ways:

1. An object $x \in U$ certainly belongs to X if all objects from the P-elementary set defined by $I_P(x)$ also belong to X. A set of all objects certainly belonging to X creates the P-lower approximation of X and can be represented as follows:

$$\underline{I_P} = \{x \in U : I_P(x) \subseteq X\} \tag{2.6}$$

2. An object $x \in U$ can possibly belong to X if at least one object from P-elementary set $I_P(x)$ can possibly belong to X. All the objects that could possibly belong to X are denoted as P-upper approximation of X, defined as:

$$\overline{I_P} = \{x \in U : I_P(x) \cap X \neq \emptyset\} \tag{2.7}$$

   Therefore the set $U - \overline{I_P}$ represents the negative region containing the set of objects that can be definitely ruled out as members of the target set X.

The tuple $< \underline{I_P}, \overline{I_P} >$ representing a lower boundary of the target X and the upper boundary of the target X creates a rough set [36]. Using above notion we can define P-boundary region which is a difference between upper and lower approximation [15].

$$BN_P(X) = \overline{I_P} - \underline{I_P} \tag{2.8}$$

The $BN_P(X)$ is a set of elements which cannot be certainly classified neither as X nor as not-X with respect to the set of attributes P. If the boundary region of X is empty then the set represented by tuple $< \underline{I_P}, \overline{I_P} >$ is crisp, otherwise we deal with inexact set which is called rough set. Until this moment we can see that rough sets concept can be defined quite generally by means of topological operations: interior and closure, called approximations. They express the knowledge about pattern in terms of granules, not by a precise measure [42].

The illustrative example of rough sets reasoning is presented in fig. 2.1

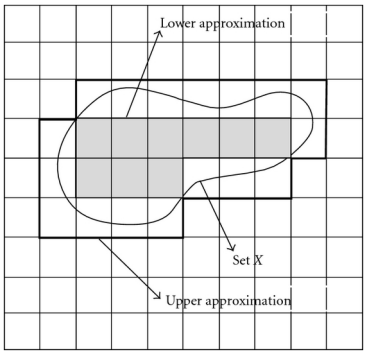

Figure 2.1: Rough sets example presenting lower and upper approximations

2.3.3 Rough sets indicators

In rough sets theory we can define few indicators. First of all with every subset of $X \subseteq U$ described by P subset of attributes we can associate an indicator called an accuracy of approximation defined as:

$$\alpha_P(X) = \frac{\underline{I_P}(X)}{\overline{I_P}(X)} \tag{2.9}$$

The grater $\alpha_P(X)$ is, the better approximation we have which means that many objects can be certainly classified as belonging to the target set X.

Another important indicator is a quality of approximation of $X \subseteq U$ by attributes from subset P. It represents the percentage of correctly classified objects using attributes P from subset X:

$$\gamma_P(X) = \frac{\overline{I_P}(X)}{|X|} \tag{2.10}$$

Assuming that we can partition U into n decision classes using P non-empty subset of attributes from C, the quality of classification can be defined as the ration of all correctly classified objects into classes:

$$\gamma_P(CLASS) = \frac{\sum_{i=1}^{n} \overline{I_P}(CL_i)}{|U|} \tag{2.11}$$

These indicators can be used in determining the quality of rough set algorithm or for finding the optimal reduct. Generally, the main goal in constructing rough sets algorithm is to obtain rule set with possible the greatest number of certain decisions.

2.3.4 Properties of rough sets

The same as classical sets, rough sets can be described by the following properties:

1. $\overline{I_P} \subseteq X \subseteq \underline{I_P}$
2. $\overline{I_P}(\emptyset) = \underline{I_P}(\emptyset) = \emptyset;\ \overline{I_P}(U) = \underline{I_P}(U) = U$
3. $\overline{I_P}(X \cup Y) = \overline{I_P}(X) \cup \overline{I_P}(Y)$
4. $\underline{I_P}(X \cap Y) = \underline{I_P}(X) \cap \underline{I_P}(Y)$
5. $\overline{I_P}(U - X) = -\underline{I_P}(X)$
6. $\underline{I_P}(U - X) = -\overline{I_P}(X)$

It is easily seen that the lower and the upper approximations of a set are interior and closure operations in a topology generated by the indiscernibility relation $I_P(X)$. Additionally, in rough sets theory we can define four types of data vagueness [42]:

- $\overline{I_P}(X) \neq \emptyset$, $\underline{I_P}(X) \neq \emptyset \; IF \; X$ is roughly I_P-definable. It means that for some elements from U we can decide whether they belong to X or $U - X$ using I_P
- $\overline{I_P}(X) \neq \emptyset$, $\underline{I_P}(X) \neq U \; IF \; X$ is internally I_P-indefinable. It means that we are able to decide which elements from U belong to $U - X$, but we don't know if they belong to X using I_P
- $\overline{I_P}(X) \neq \emptyset$, $\overline{I_P}(X) = U \; IF \; X$ is externally I_P-definable. It means that we are able to decide which elements from U belong to X, but we don't know if they belong to $U - X$ using I_P
- $\overline{I_P}(X) \neq \emptyset$, $\overline{I_P}(X) = U \; IF \; X$ is totally I_P-indefinable. It means that we are unable to decide for any element from U if it belongs to X or $-X$ using I_P

2.3.5 Rough sets reasoning from data

The category description can be done in two ways:

1. extensional
2. intentional

Each of these approaches differs in the way how pattern from dataset is represented. To represent a concept we have to be able to identify all objects belonging to this category. With the former approach we have no insight into decision engine so we do not know how to assign new objects to the category. In the latter approach we represent the category based on the set of rules. The same approach is done in rough sets algorithm where an elementary granules (concepts) of knowledge build blocks consisting of indiscernible pattern from the universe of discourse U. As stated in section 2.3.2 reconsidered dataset or problem can be represented as a table. When decision attributes from D are added to the table it is transformed into decision table T where each row is associated with decision rule.

In this section a practical example will be presented to clear all the things out. As an example let reconsider well-known problem of patients suffering from flu [41], [42]. The task is to decide if a patient described by 3 attributes is ill or not. Table 2.1 represents an information system IS about healthy patient and those suffering from flu. Attributes: *Headache*, *Muscle-pain*, *Temperature* are called condition attributes and denoted as $(c_i, i \in (1, \ldots, q))$, while the attribute *Flu* (last column in table 2.1) is considered as a decision attribute c_d.

Table 2.1: Example dataset showing healthy patients and suffering from flu

Patient	Headache c_1	Muscle pain c_2	Temperature c_3	Flu c_d
p1	no	yes	high	yes
p2	yes	no	high	yes
p3	yes	yes	very high	yes
p4	no	yes	normal	no
p5	yes	no	high	no
p6	no	yes	very high	yes

Each row of a decision table determines a decision rule.
Rule 1: IF c_1 IS 'NO' AND c_2 IS 'YES' AND c_3 IS 'HIGH' THEN c_d IS 'YES'
Rule 2: IF c_1 IS 'YES' AND c_2 IS 'NO' AND c_3 IS 'HIGH' THEN c_d IS 'YES'
Rule 3: IF c_1 IS 'YES' AND c_2 IS 'YES' AND c_3 IS 'VERY HIGH' THEN c_d IS 'YES'
Rule 4: IF c_1 IS 'NO' AND c_2 IS 'YES' AND c_3 IS 'NORMAL' THEN c_d IS 'NO'
Rule 5: IF c_1 IS 'YES' AND c_2 IS 'NO' AND c_3 IS 'HIGH' THEN c_d IS 'NO'
Rule 6: IF c_1 IS 'NO' AND c_2 YES 'NO' AND c_3 IS 'VERY HIGH' THEN c_d IS 'YES'

After analyzing table 2.1 it is noticeable that some patients cannot be distinguished between each other, for example $p2$ and $p5$. It is possible to generate different indiscernible relations based on the chosen attributes. In case of *Headache* attribute patients: $p2$, $p3$, $p5$ are indiscernible; patients $p2$, $p5$ are indiscernible with respect to attributes *Headache*, *Muscle-pain* and *Temperature*, or if we use *Headache* and *Muscle-Pain* the set U can be divided into three sets: {$p1$, $p4$, $p6$}, {$p2$, $p5$}, {$p3$}.

Now it is time for defining key features of rough sets. Over the table 2.1 we can define two concepts (target sets): *Flu* and *Not Flu*. For the first concept the lower approximation set of patient certainly having flu is {$p1$, $p3$, $p6$}, while the upper approximation of patients possibly suffering from flu is {$p1$, $p2$, $p3$, $p5$, $p6$}. The boundary region for concept *Flu* is a set of {$p2$, $p5$} patients. For the concept *Not Flu* the lower approximation is the set {$p4$}, whereas the upper approximation is the set {$p2$, $p4$, $p5$}. Again the boundary region is the set {$p2$, $p5$}.

Additionally, we can measure the accuracy of approximation $\alpha_P(x)$ for each concept. This can indicate if the set of attributes used for describing the concept is correctly chosen. For the *Flu* concept where X={$p1$, $p2$, $p3$, $p6$} is described by set of attributes P={*Headache*, *Muscle-pain*, *Temperature*} the accuracy of approximation is:

$$\alpha_P(Flu) = \frac{3}{5}$$

On the other hand when we take only one attribute P={*Temperature*}, then we get lower approximation of {$p3$, $p6$} and upper approximation of {$p1$, $p2$, $p3$, $p5$, $p6$ } resulting in:

$$\alpha_P(Flu) = \frac{2}{5}$$

To sum up, $\alpha_P(x)$ is a very important indicator in rough sets theory and tells which attributes better characterize target concept. This example has shown that because of the granule representation of knowledge in rough sets approach some object cannot be discerned. It is very important to choose proper condition attributes because this determines how upper and lower approximations are represented by objects from U.

2.4 Fuzzy logic

2.4.1 Introduction

In fuzzy logic an element membership to a set is described by a membership function which assigns value from interval $[0, 1]$:

$$\mu_A : U \to [0, 1],\ A \subseteq U \tag{2.12}$$

The properties of membership function μ are as follows:

- $\mu_\emptyset(x) = 0,\ x \in U$
- $\mu_U(x) = 1,\ x \in U$
- $\mu_X(x) \leq \mu_Y(x),\ IFX \subseteq Y$
- $\mu_{\overline{A}}(x) = 1 - \mu_A(x),\ x \in U$

where $\emptyset$ is an empty set and $\overline{A}$ is the complement of the set A.

Fuzzy logic is a superset of Boolean logic that has been extended to handle the concept of partial truth - values between "completely true" and "completely false" [7], [13]. This theory was introduced by Dr. Lotfi Zadeh in the 1960's as a tool for modelling the uncertainty of natural language. This approach can be applied to many problems, analyze only few examples:

- Days of a week- there is a question which day is mostly associated with a weekend. Generally, we would assume that the membership value g associated with each day will be as follows: {Monday=0, Tuesday=0.2, Wednesday=0.4, Thursday=0.5, Friday=0.7, Saturday=1.0, Sunday=0.9}
- The seasons- there is not evident boundary when one season stops and another starts. This example can be illustrated quite precisely by fig. 2.2.

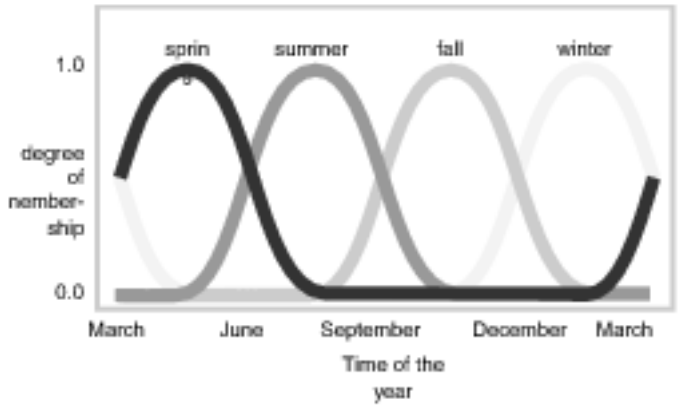

Figure 2.2: Example of how fuzzy logic can be used to describe the season of the year

- The height: small and tall definitions- from the group of people it is hard to define who is tall and who is small interchangeably. To solve this problem we can define the concept *Tall* and assign value to each person describing its value. This process is presented in fig. 2.3

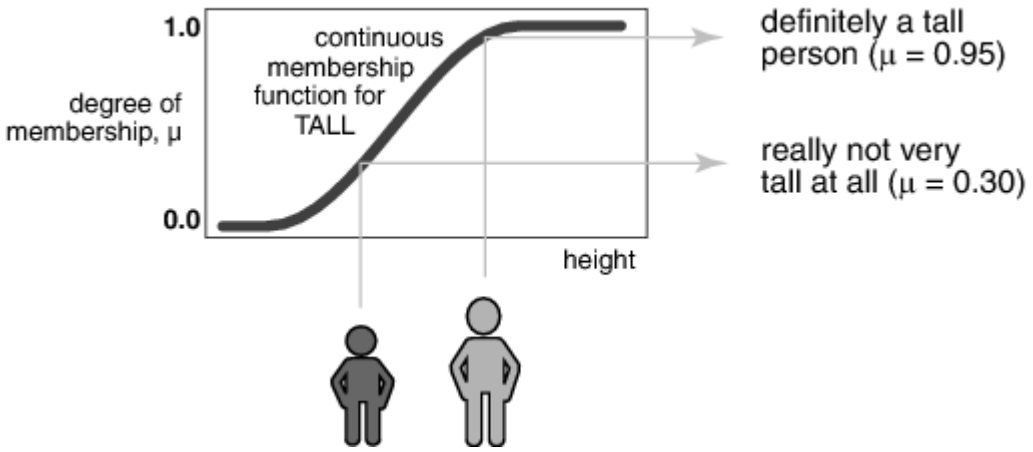

Figure 2.3: Fuzzy approach for defining person height

As in the section 2.3.1, let reconsider the problem of defining if person is small or tall more deeply. First of all, we have to define a fuzzy set (concept) *Tall* which

will answer the question "to what degree is person x tall?". Zadeh describes *Tall* as a linguistic variable, which represents our cognitive category of "tallness" [26]. To each person in the universe of discourse, we have to assign a degree of membership in the fuzzy subset *Tall* based on the membership function (for example the same as presented in fig. 2.3). Table 2.2 shows an example of fuzzy reasoning:

Table 2.2: Table describing how person is tall by the fuzzy logic linguistic variable

Person	Height [m]	degree of tallness
p1	1.5	0.0
p2	1.6	0.2
p3	1.7	0.4
p4	1.8	0.5
p5	1.9	0.6
p6	2.0	1.0

Fuzzy numbers are fuzzy subsets generated over the attribute domain. They have a peak or plateau with membership grade 1, over which the members of the universe are completely in the set. The membership function is increasing towards the peak and decreasing away from it. There are different types of membership functions and their usage strongly depends on the type of reconsidered problem. One of the most commonly used in the literature are: triangular, trapezoidal, Gaussian shapes (see example in fig. 2.5).

2.4.2 Fuzzy reasoning from data

Reasoning in fuzzy logic is based on decision rules the same as in rough sets approach [12]. Rules are expressed in the form of IF *COND* THEN *DECISION* which can be divided into antecedent set ($A_{ri},\ i \in (1,\ldots,q)$) and one consequent determining the output of the rule. In the pattern recognition task when we deal with patterns described by d features rules have the form presented by eq. (2.13)

$$IF\, x_1 = A_{r1}\, AND\, x_2 = A_{r2}\, AND\, \ldots\, AND\, x_d = A_{rd}\, THEN\, class\, C_r\, with\, CF_r \tag{2.13}$$

The *AND*, *OR*, and *NOT* operators of Boolean logic exist in fuzzy logic and are usually defined as the minimum, maximum, and complement. When they are defined this way, they are called the Zadeh operators [3].

Fuzzy logic classification is based on three main steps:

1. Fuzzyfication- in the fuzzyfication process we convert continuous quantity into fuzzy number based on the appropriate membership function value. It requires defining membership grade of crisp input x in the fuzzy set.

2. Rule induction- there are different types of fuzzy inference systems, but one of the most commonly used (the same as in this paper) is Mamdani inference system [48].

3. Deffuzification- the process of producing a quantifiable result in fuzzy logic from given fuzzy sets and corresponding membership degrees.

The whole process of fuzzy reasoning is presented in fig. 2.4

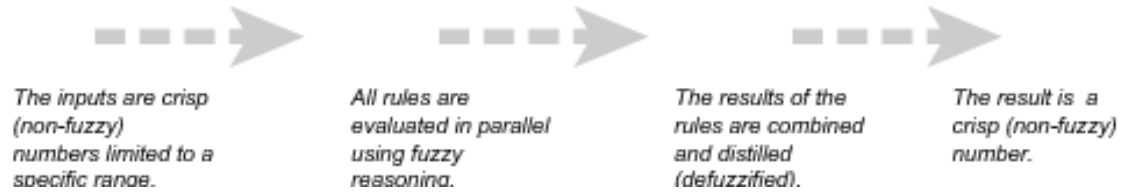

Figure 2.4: Steps applied in fuzzy reasoning.

Let reconsider a simple example, which should clear all ambiguities (the following example is based on [1]). The problem is connected with estimating the level of risk involved in software engineering project. There are two input to the system (funding, staffing) and one output (risk). Membership functions are represented as triangular shapes (fig. 2.5)

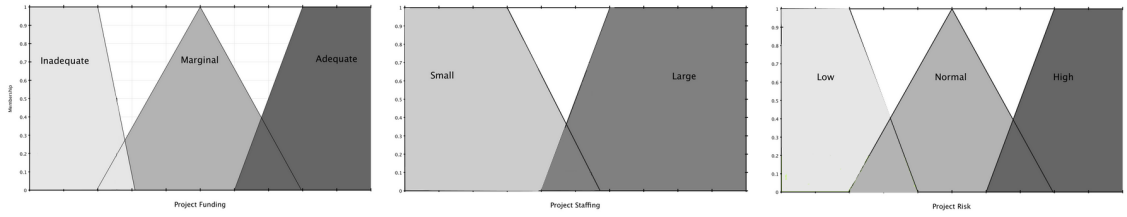

Figure 2.5: Input and output fuzzy linguistic variables for the described system.

The next important thing in fuzzy logic beside fuzzy values are fuzzy rules. In this problem they are provide a priori and are in the following way:

Rule 1: IF funding IS adequate OR funding IS small THEN risk IS low
Rule 2: IF funding IS marginal AND staffing IS large THEN risk IS normal
Rule 3: IF funding IS inadequate THEN risk IS high

Let say that we want to calculate project risk for inputs: funding $= 35$ and staffing $= 60$.

1. The first step is to convert crisp values into fuzzy representation. Activating each input membership functions fuzzy values are as follows:

$$\begin{array}{l} input\ 1 \\ \hline \mu_{inadequate}(35) = 0.5 \\ \mu_{marginal}(35) = 0.2 \\ \mu_{adequate}(35) = 0.0 \end{array}$$

$$\begin{array}{l} input\ 2 \\ \hline \mu_{small}(60) = 0.1 \\ \mu_{large}(60) = 0.7 \end{array}$$

2. Now it is time f or rule induction where OR is treated as a max operator, AND as min operator.

$$\begin{aligned} Rule\,1: &\quad \mu_{low} = 0.0 + 0.1 = 0.1 \\ Rule\,2: &\quad \mu_{normal} = 0.2 \cdot 0.7 = 0.14 \\ Rule\,3: &\quad \mu_{high} = 0.5 \end{aligned}$$

3. After performing clipping of consequent membership functions for each rule (example given in fig. 2.6), the final crisp output can be calculated. It is done by defuzzification method, where one of the most commonly used is a centroid formula given by eq. (2.14) [1], [2].

$$CO = \frac{\sum_{x=a}^{b} \mu_A(\chi) \cdot x}{\sum_{x=a}^{b} \mu_A(\chi)} \tag{2.14}$$

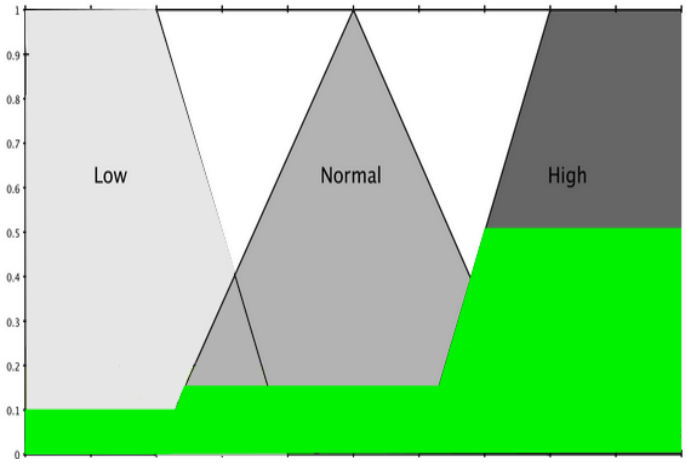

Figure 2.6: Example of clipping the consequent membership function as the result of rule induction

Using eq. (2.14) the output of the system is as follows:

$$CO = \frac{(0+10+20)\cdot 0.1 + (30+40+50+60)\cdot 0.14 + (70+80+90+100)\cdot 0.5}{0.1\cdot 3 + 0.14\cdot 4 + 0.5\cdot 4} = 69.3$$

It means that for input values funding $= 35$ and staffing $= 60$ the project risk is about 69%.

2.4.3 Genetic-based machine learning approaches

In the literature one can find many examples of fuzzy logic applications [4], [10]. Generally, there are two main approaches:

1. some expert knowledge is available and fuzzy rules are created beforehand.
2. no knowledge about data set is present so some techniques of data mining must be applied to extract rules from the training set.

At this point it must be strongly emphasized that in this thesis the main focus is based on fuzzy algorithm for pattern recognition task where there is no prior knowledge about fuzzy system and decision rules. There are different approaches for construction fuzzy logic algorithm from raw dataset, for example optimization techniques such as gradient descent or heuristic algorithms [9], [17], [30]. In this paper the genetic algorithm is proposed for creating an optimal rule set.

One can wonder what is the purpose of applying genetic algorithm into rule generation. For the simplicity let reconsider such a case. N training patterns are available and nothing more. Now the following question arises: how to properly divide the feature space into fuzzy set, and how to generate decision rules when there is no expert knowledge. The simplest and straightforward solution is to create let say 10 triangular membership functions for each attribute, and check each possible combination. Is this approach optimal? The efficiency of this solution strongly depends on the complexity of the problem. For d-dimensional feature space and k fuzzy membership function per each attribute the number of possible combination for generating one rule is equal to k^d. For greater d (for example wine dataset from *UCI* repository has 13 attributes) it is impossible to find the proper combination in a reasonable time. Here is an open spot for genetic algorithm to show its searching abilities.

There are two main methods of genetic-based machine learning approaches [32], [33]:

1. Michigan template- it is a population of fuzzy rules and a single fuzzy rule is handled as an individual (see fig. 2.7). The evaluation of each fuzzy rule is performed by classifying all the given training patterns by the available rule

set N_{rule}. At the end of each iteration new individuals are created through genetic operators and merged to the current population. For the next generation N_{rule} best individuals are taken. The whole procedure can be summarized as follows:

(a) Generate N_{rule} fuzzy rules

(b) Evaluate the fitness of each fuzzy rule in the current population

(c) Generate $N_{replace}$ fuzzy rules using genetic operators

(d) Merge $N_{replace}$ fuzzy rules with current population and choose the best N_{rule} individuals for the next generation

(e) Return to point (b) is stopping condition is not fulfilled (number of generations)

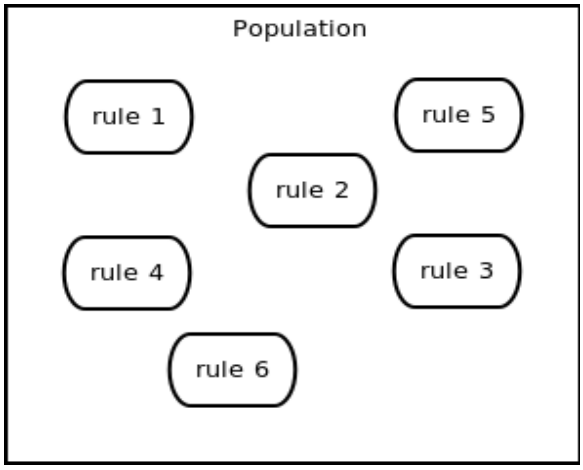

Figure 2.7: Example of Michigan approach for constructing Genetic Fuzzy logic algorithm

2. Pittsburgh template- in this approach a set of fuzzy rules is handled as an individual (see fig. 2). In this case the length of a single individual is equal to $n \cdot N_{rule}$, where n is the length of single fuzzy rule. Algorithm starts with N_{pop} randomly generated rule sets. The fitness value of a single individual is the number of correctly classified patterns in the training set by a given rule set. The procedure for algorithm is as follows:

(a) Generate N_{pop} individuals consisting of N_{rule} fuzzy rules each

(b) Calculate the fitness value of each rule set (individual)

(c) Generate $N_{replace}$ new rule set using genetic operators.

(d) Merge $N_{replace}$ fuzzy rule sets with current population and choose the best N_{pop} individuals for the next generation

(e) Return to point (b) is stopping condition is not fulfilled (number of generations)

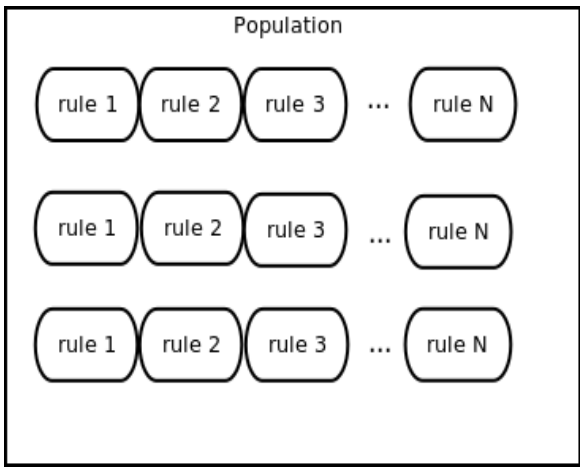

Figure 2.8: Example of Pittsburgh approach for constructing Genetic Fuzzy logic algorithm

In this thesis the first approach will be implemented which means that individual is modelled as a single rule.

2.5 Attribute reduction

Many problem are complex and multidimensional. For example Sonar dataset from *UCI* Repository has 60 attributes describing a single pattern. Usually we hope to recognize pattern in a relatively lower dimensional to reduce cost in measuring and processing information and enhance the interpretability of learned models. Feature selection or reduction is done for classifiers to remove the noise and superfluous data. Generally, this is not an easy task and requires a lot of computation [2], [6].

A reduct is a set of attributes that ensures the same classification of elements from U as the rudimentary set of attributes. More than one reduct can exist for one information system. The core of attributes is the set of attributes from Q that all the attributes are indispensable. An attribute is dispensable if the following criterion is fulfilled:

$$I(P) = I(P - a), \text{ for } \{a\} \in P \subseteq Q$$

In the literature there are many examples of how to apply attribute reduction

- information measures
- distance measures
- dependence measures
- consistency measures

Different approaches provide various results and their application strictly depend on the type of data. One of the best known methods are Principal Component Analysis, Factor analysis or optimization and heuristic techniques. In the recent years heuristic techniques are very often used because of their speed and searching abilities. A lot of work has been devoted for feature extraction and this is not the intention to explore this area of research. Basing on the results conclusions presented in the literature, in this thesis a genetic algorithm is proposed for extracting valuable features from the set of all attributes [29], [50].

2.6 Genetic algorithm

Genetic Algorithm is an element of evolutionary computation, which is a rapidly growing area of soft computing [44], [45]. GA is based on the principles of natural selection and genetic modification. As optimization methods, GA operates on a population of points, designated as individuals. Each individual of the population represents a possible solution of the optimization problem. Individuals are evaluated depending upon their fitness which indicates how well an individual of the population solves the optimization problem. To sum up, GA has the following general features:

1. GA operates with a population of possible solutions (individuals) instead of a single individual. Thus, the searching process can be carried out in a parallel form or sequentially.
2. GA is able to find the optimal or sub-optimal solutions in complex and large search spaces. Moreover, it can be applied to nonlinear optimization problems with constraints defined in discrete or continuous search spaces.
3. GA examines many possible solutions at the same time, so there is a higher probability that the search process can converge to an optimal solution.

There are four main parts in each GA process to reconsider (graphically presented as a flow chart 2.9):

1. the problem representation or encoding
2. fitness or objective function definition

3. fitness-based selection

4. evolutionary reproduction of candidate solutions (individuals or chromosomes).

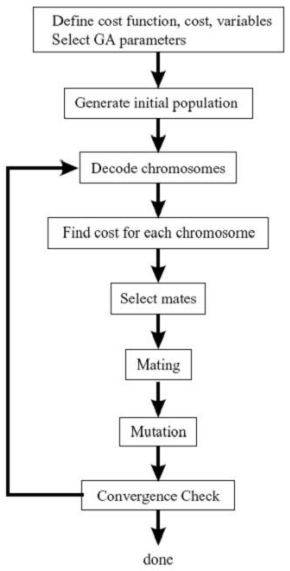

Figure 2.9: Diagram representing phases in genetic algorithm evaluation

Genetic algorithms are widely used as a search techniques in the various fields. In this thesis it will be used for finding an optimal cuts in the attribute space and to apply attribute reduction. The success of a genetic algorithm can be quantified by estimating the cost, time required and the quality of final obtained solution.

In the literature there can be found many examples of how GA is useful in solving hard optimizations problems, but beside unquestionable advantages there also exist downsides. A traditional GA without any diversity maintenance mechanism often suffers from getting stuck on the suboptimal peaks, because almost the entire GA population would have converged to a single peak, as a result of the rapid loss of population diversity [46], [47].

There is a great deal of work showing how to set the optimal parameters in an evolutionary algorithm to obtain required speedup and solution accuracy, but this is not the main issue in this thesis. For more exact information see [44], [45].

When designing genetic algorithm one of the most important things to ensure proper crossover and mutation operations. More precise information about these operators will be presented in section 3.

2.7 Hybrid classifiers

In the recent years, there is an increasing interest in methods of combining multiple learning systems into hybrid one [6], [7]. The main advantage of such approach is its ability to find different explanation for the dataset for each classifier. If classifiers make errors on different parts of the feature space it is possible that the ensemble of classifiers will complement each other and the final classification will be better.

Generally, there are two types of hybrid classifiers (example presented in fig. 2.10):

1. multiexpert systems- classifiers work in parallel, each of them is trained and tested on the same data and independent decisions are combined to compute the final result. The most common example is a majority voting.

2. multistage systems- classifiers are connected in a sequence where the next classifier is trained and used for classification only if the previous classifier rejected the pattern.

It is hard to determine which approach is the best. Each system has its pros and cons and the choice depends of the type of dataset and available classifiers. In this thesis the second approach is implemented where the first segment is built by rough sets classifier and the second one is constructed by fuzzy logic.

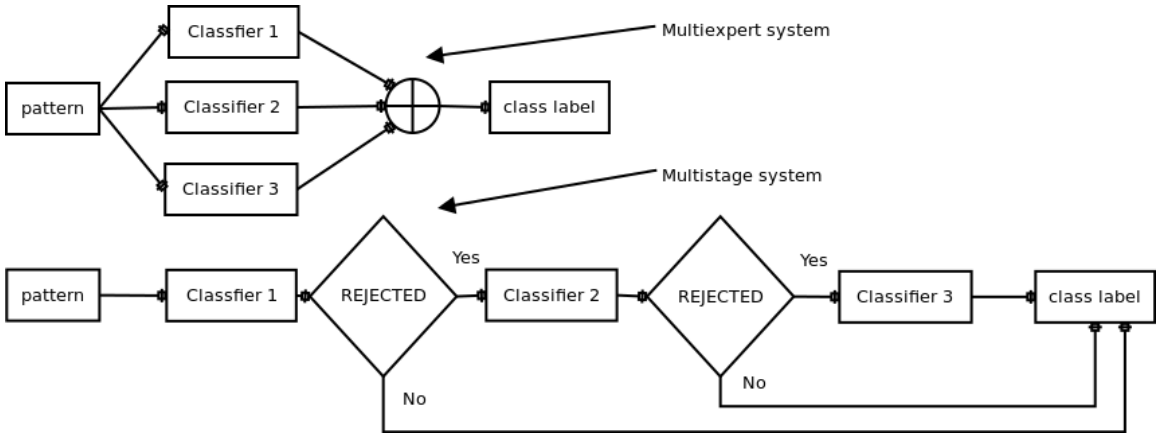

Figure 2.10: Example of two approaches for constructing hybrid classifiers

3 Algorithm construction

In this section basic information about algorithm construction will be presented. For the classification task four algorithms were implemented and com-

pared.

At the beginning, we start with the basic algorithm of rough sets, later present rough sets algorithm with modification of decision rules and at last the multistage hybrid algorithm consisting of genetic algorithm, rough sets and fuzzy logic is shown in greater details.

3.1 Rough sets algorithm construction

The basic rough sets algorithm with constant step of granulation can be summarized in six steps [36], [37]:

1. If the attributes are the real numbers then the granulation preprocessing is needed first. To calculate the proper interval eq. (3.1) is used:

$$v_{pi}^{i} = \frac{f_i}{G} \cdot (f_{i,max} - f_{i,min}) \tag{3.1}$$

   where v_{pi}^{i} is the interval label for i-th attribute; f_i is the i-th real attribute value for pattern x, $f_{i,min}$ and $f_{i,max}$ denote the extend of the i-th feature calculated in the training phase. If for any attribute from x in the testing phase $f_i > f_{i,max}$ then $v_{pi}^{i} = G - 1$ or $f_i < f_{i,min}$ then $v_{pi}^{i} = 0$, in other cases eq. (3.1) is applied.

   After this step, the value of each attribute is represented by the number of interval in which this attribute is included. For each attribute from $l = (1, \ldots, q)$ we choose the same numbers of intervals K_l called step of granulation G. For the l-th attribute denoted by $v_{p_l}^{l}$ it is defined its p_l interval from $p_l = (1, \ldots, K_l)$

2. Using training dataset construct the decision table T where each row represents a pattern. Over the table T define the set $FOR(C)$ of decision rules of the following form:

$$IF\,(x_1 = v_{p_1}^{1})\,AND\,(x_2 = v_{p_2}^{2})\,AND\,\ldots\,(x_q = v_{p_q}^{q})\,THEN\,\Psi(S, x) = j$$

   Each generated rule is evaluated and the strength factor is assigned to it determining the accuracy of approximation (see section 2.3.3)

3. For the created set of formulas $FOR(C)$ for each $j = 1, \ldots, m$ an algorithm calculates lower $\underline{I_P}$, upper $\overline{I_P}$ approximations and the boundary region BN_P which are determined by the set of conditional attributes $P \subseteq Q$

4. In order to classify new pattern x algorithm looks for matching rules in the set $FOR(C)$ (rule is activated if the left condition is fulfilled by attributes describing x).

5. If there is only one matching rule r, then the pattern x is classified to the class which is indicated by r decision attribute j, because for sure such a rule is belonging to the lower approximation of all rules indicating j. Rule r is denoted as a *certain* rule.

6. If there is more than one matching rule in the set $For(C)$, it means that the recognized pattern should be classified by the rules from the boundary regions and in this case as a decision algorithm takes the index of boundary region for which the strength of corresponding rule is maximal. All activated rules are denoted as *possible* rules.

7. In other cases: no appropriate rule was found or few rules have the same strength factor then the unknown pattern x is rejected.

3.2 Rough sets algorithm construction with modification of decision rules

It can happen that for certain number of intervals algorithm cannot find patterns in the training set, so as the consequence dummy rule are generated, useless in the classification process (the strength of the rule is 0). The main drawback of algorithm presented in section 3.1 is the fact that it starts with an arbitrary chosen step of granulation and its accuracy strongly depends on it. In this section the recursive modification of the previous algorithm is presented allowing for automatically changing the step of granulation is the pattern x is rejected. The modification is as follows [38]:

1. Algorithm starts with an arbitrary chosen step of granulation G, generally it is a high value to ensure that recursion can be invoked by decreasing G. Shortly speaking, we divide every domain of feature into G intervals. At first algorithm repeats the whole procedure 1-6 described in the previous section.

2. If for the pattern x algorithm cannot find neither certain nor possible decision rule, it means that there is no proper representation in the learning set. In such a situation algorithm tries to find matching rule by decreasing recursively the current interval G by factor $\epsilon = 1$ for every condition attribute $l = 1, \ldots, q$ until the proper rule is found. If *certain* rule or *possible* rules with maximum strength are induced then the algorithm returns decision attribute j, otherwise if for every attribute $G = 1$, then the pattern x is rejected.

The recursion is time consuming so to enhance the process of finding the proper decision formulas for different G the decision set $FOR(C)$ are stored in the memory for faster retrieval.

3.3 Fuzzy logic algorithm construction

3.3.1 Problem formulation

In the fuzzy logic algorithm one of the most important key is creation of rules which will ensure proper classification. When there is no expert knowledge about dataset it is not an easy task to generate them from scratch. In the literature one can find many practical examples of how to generate *IF-THEN* rules, from the statistical tools to heuristic algorithms. In the recent the most appealing and effective approach is genetic algorithm [5], [14], [24]

For the algorithm construction we have to make few assumptions:

- For the learning procedure N training patterns are available
- A set F of linguistic values and their membership functions is given for describing each attribute.

The second point is the most important and the choice of the proper membership functions affects the classification accuracy [18], [7]. It requires in-depth explanation of how set F is generated in this thesis, how to partition each attribute into linguistic values and how to describe each membership function. Genetic algorithm is used here for finding an optimal parameter settings.

Fig. 3.1 shows an example of how to generate fuzzy set F of possible membership functions. In this example 14 membership functions are used. Each function has a subscript defining its linguistic value and a proper location in the feature space. To emphasize the complexity of the problem let take into account that each attribute is divided in the same way as depicted in fig. 3.1. Having d-dimensional feature space, the number of all possible combinations of membership functions for rule construction is equal to 14^d. It is impossible to evaluate each potential solution it is in a reasonable time [29], [35].

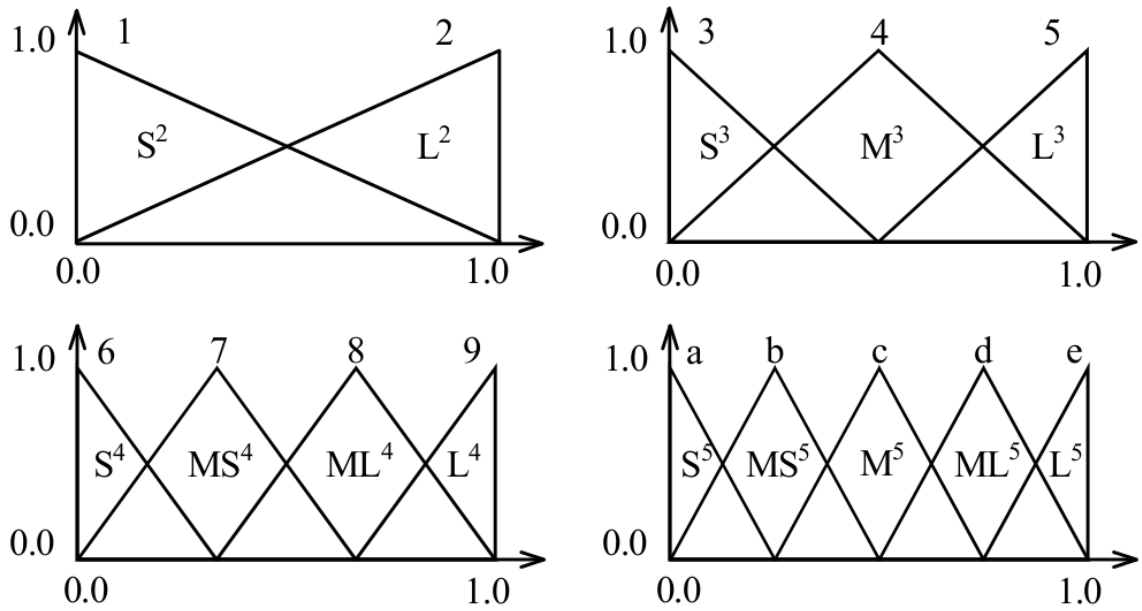

Figure 3.1: Example fuzzy partition set for an attribute

For classification task single fuzzy rule is in the following form [32], [21]:

$$R_r : IF\, x_1 = A_{r1}\, AND\, x_2 = A_{r2}\, AND\, \ldots\, AND\, x_d = A_{rd}\, THEN\, class\, C_r\, with\, CF_r$$

where $x = (x_1, x_2, \ldots, x_d)$ is a d-dimensional pattern vector; A_{ri} is an antecedent fuzzy set with a linguistic label (taking into account the example from 3.1 A_{ri} can have one label from the set $\{1, 2, \ldots, 9, a, b, \ldots, e\}$); C_r is a consequent fuzzy set determining the class label $\{1, \ldots, m\}$, CF_r is a rule strength; r denotes the number of the rule from the set of all possible rules $\{1, \ldots, N_{rule}\}$. Generally, N_{rule} is about ten or twenty.

3.3.2 Rule generation

The process of rule generation for fuzzy logic without the expert knowledge is complex and done in couple steps. At first, using available training set, generate randomly N_{rule} rules. For each training pattern x_p calculate the compatibility grade of a single rule connected with antecedent part $A_r = (A_{r1}, A_{r2}, \ldots, A_{rd})$ using the product operator of each membership function $\mu_{A_{ri}}$ determined for A_{ri}:

$$\mu_{A_r}(x_p) = \mu_{A_r1}(x_p) \cdot \mu_{A_r2}(x_p) \cdot \ldots \mu_{A_rd}(x_p) \tag{3.2}$$

If we know how to calculate the compatibility grade of each training pattern now we can determine C_r and CF_r for each rule. The fuzzy probability $P(class\, j|A_r)$ of

class j, $j = (1, \ldots, m)$ is given by eq. (3.3)

$$Pr(class\, j|A_r) = \frac{\sum\limits_{x_p \in class\, j} \mu_{A_r}(x_p)}{\sum\limits_{p=1}^{m} \mu_{A_r}(x_p)} \tag{3.3}$$

For the r-th rule R_r the label of class is assigned according to the winning rule, which means that the label with maximal probability is chosen:

$$R_r : C_r = max_{j=\{1,\ldots,m\}}\{Pr(class\, j|A_r)\} \tag{3.4}$$

In the learning phase it can happen that rule R_r can be activated by patterns coming from different classes. To ensure the proper classification, each rule has a strength factor which tells how precisely rule R_r predicts the consequent class h.

$$R_r : CF_r = Pr(class\, j|A_r) - \sum_{j=1, j\neq C_r}^{M} Pr(class\, j|A_r) \tag{3.5}$$

If CF_r in eq. (3.5) is negative then rule R_r is denoted as *dummy* and is not taken for further reasoning, otherwise it is used in defuzzification process to determine the final class label [19], [27].

3.3.3 Fuzzy reasoning

Let assume that N_{rule} fuzzy rules are generated with indicators C_r, CF_r determined by eq. (3.4), (3.5). Then the process of classification is done as follows:

$$\Psi(S, x_p) = C_q \leftarrow max_{j=\{1,\ldots,M\}}\{\mu_{A_q}(x_p) \cdot CF_r\} \tag{3.6}$$

The label of the class for unknown pattern is determined by a winner rule R_w that has the maximum compatibility grade and the rule strength CF_r.

If multiple fuzzy rules have the same maximum product μ_{A_r} but different consequent classes then the classification is rejected. The same action is taken if no fuzzy rule is compatible with the incoming pattern x_p.

3.3.4 Genetic algorithm for fuzzy algorithm construction

In this section genetic algorithm will be described in greater details. This algorithm was used to generate initial number N_{rule} of fuzzy rules for classification. Basic assumptions:

- Fuzzy rule encoding is the same as presented in section 3.3.1

- Training data set is given with N patterns
- Triangular membership functions are used and described by 2-tuple (a, b), where a is the center value (where $\mu(x) = 1$), and b determines left and right extend of membership function, respectively.

$$\mu(x) = \begin{cases} \frac{-1}{b} \cdot x + \frac{a+b}{b} & x \geq a \, and \, x \leq (a+b) \\ \frac{1}{b} \cdot x - \frac{a-b}{b} & x \geq (a-b) \, and \, x < a \\ 0 & otherwise \end{cases} \tag{3.7}$$

- Possible partitions of the feature space are determined in the same way as in the example presented in fig. 3.1
- Genetic algorithm uses standard operations such as cross-over, mutation, population generation, fitness evaluation.
- As the template for genetic fuzzy algorithm Michigan approach is used which means that we have N_{rule} number of individuals in the population

Next few step will present the whole structure of genetic algorithm used in this section:

- Chromosome representation and encoding:
  - Each individual represents a single fuzzy rule R_r from the set of containing N_{rule} rules. The length of the chromosome is the same as the number of attributes describing the pattern x. Each allele has value determining which linguistic variable is used in the current rule. Reconsider Iris dataset which is a 4-dimensional classification problem where for each attribute 14 membership functions plus one variable telling to omit the attribute (called $DON'T\,USE$) are generated. An exemplary individual can be as follows:

    $$1|c|DON'T\ USE|4||1|0.85$$

    Above individual can be decoded into rule presented in fig. 3.2

Figure 3.2: Example rule decoded from the individual chromosome (attribute x_3 was omitted in this rule)

- Individual evaluation

- To ensure proper genetic algorithm process an appropriate fitness function must be defined [11], [22]. Firstly the nature of pattern recognition task must be taken into account and secondly the structure of the fuzzy algorithm. Generally, the main goal is to generate rules with the highest CF_r grade, the smallest number of attributes and the highest classification rate. Fitness function is given by eq. (3.8)

$$F_{fg} = w_1 \cdot NC + w_2 \cdot NNC + (\frac{1}{NOF})^2 + w_3 \cdot CF \tag{3.8}$$

  where w_1, w_2 are weights for a reward and punishment to the rule based on the classification result (in simulations $w_1 = 5$, $w_2 = 10$); NC and NNC are the numbers of correctly recognized and misclassified patterns by a particular rule, respectively; NOF is the number of attributes used by the rule (in the above example $NOF = 3$); CF is the strength factor of the rule and w_3 is the weight (in the simulations $w_3 = 10$). The best individuals are those which maximize function F_{fg}

- Cross-over, mutation and population generation
  - From the whole population two individuals are chosen randomly to constitute father and mother parents. With a probability of 0.5 each allele is picked either from mother or father chromosome. In this way two new individuals are generated (see example in fig. 3.3)

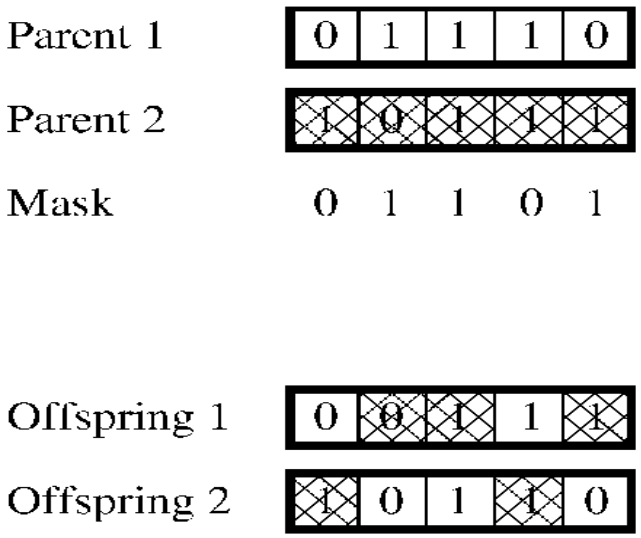

Figure 3.3: Cross-over operation used in genetic algorithm

- In the particular generation one chromosome is chosen randomly and later in each allele new membership function is taken from other possible functions. For example if in the first allele the first membership function is chosen the set of candidates is given by $\{2, \ldots, 9, a, \ldots, e, DON'T\,USE\}$
- In genetic algorithm one of the most important issue is how to generate next population [23]. Here, after the end of one generation individuals from the population are merged with those created through cross-over and mutation operations. Later, the average fitness value F_{avg} is calculated in the whole set. To the next generation those individuals are passed which their fitness indicator F_{fg} is greater than the average $F_{fg} \geq F_{avg}$

Of course it can happen that for cross-over and mutation operator newly-generated individual will be invalid (the whole chromosome contains only $DON'T\ USE$ linguistic variables). In such a situation rule is rejected and the whole generation process is repeated again.

Table 3.1 presents basic genetic algorithm parameters. Optimal values were determined during simulations by trial and error method.

Table 3.1: Parameter settings for genetic algorithm used in fuzzy logic

Parameter	value
N_{rule}	10
$N_{replace}$	$N_{rule}/2$
Crossover probability	0.9
Mutation probability	0.3
Generations	500

3.4 Multistage hybrid algorithm construction

3.4.1 Motivations

When we deal with complex data it can happen that a single classifier is not sufficient. There arises a question if connection of different classifier will improve the classification? In this thesis the hybridization of rough sets and fuzzy logic is proposed and investigated. The next subsections show algorithm construction.

3.4.2 Rough sets and genetic algorithm

Rough sets algorithm presented in section 3.1 uses an arbitrary chosen step of granulation and each attribute has the same granulation intervals. In some cases

this approach gives good results in more complex problems algorithm efficiency is low [25], [31]. Additionally, the basic rough set algorithm uses all attributes for rule construction.

Finding the optimal attribute reduct and rough set partition is NP problem [20], [2]. To overcome this obstacle a genetic algorithm is used in the similar way as in section 3.3.4. Now, a single individual describes the partition for each attribute independently. Reconsider individual encoding for 4-dimensional Iris dataset. The number of granulation intervals for each attribute is chosen from the set $\{1, 2, \ldots, K_{max}\}$, where K_{max} is the maximum value of discretization. Additionally, $DON'T\ USE$ variable is used to determine that a given attribute is excluded from the rule. The example of individual is given below:

$$|2|DON'T\ USE|K_{max}|3||120$$

It means that the first feature is divided into two intervals, the second is not used and the third and fourth are discretized into K_{max} and 3 intervals, respectively. The fitness indicator of this individual is 120. To evaluate individual the following fitness function given by eq. (3.9) is proposed:

$$F_{rg} = w_1 \cdot NC + w_2 \cdot NNC + (\frac{1}{NOF}) + w_3 \cdot (\frac{1}{NOCR})^2 \tag{3.9}$$

where w_1, w_2 are weights for a reward and punishment to the individual on the classification result ($w1 = 5$, $w_2 = 10$); NC and NNC are the numbers of correctly recognized and misclassified patterns; NOF is the number of attributes used by the rule (in the example above $NOF = 3$); $NOCR$ is the number of *certain* rules which are derived from partition given by the particular individual; w_3 is the weight (in the simulations $w_3 = 10$).

The whole procedure of constructing genetic rough sets algorithm can be summarized in few steps:

1. Determine the maximum partition value for each attribute K_{max}. In this thesis K_{max} is the same for all features.

2. Generate N_{pop} individuals by randomly assigning value from the set $\{1, 2, \ldots, K_{max}, DON'T\ USE\}$ to each allele in the chromosome.

3. Treat each individual as a rough set partition and calculate lower, upper approximations and the boundary region. Evaluate individual using fitness function F_{rg}.

4. Generate $N_{replace}$ individuals using genetic operators and merge with the current population.

5. Choose N_{pop} individual for the next generation

6. If stopping criteria is not fulfilled go to 2.

Cross-over and mutation operations are done in the same way as presented in section 3.3.4. Parameters for genetic algorithm used in this section are presented in table 3.2:

Table 3.2: Parameter settings for genetic algorithm used in rough sets

Parameter	value
N_{pop}	10
$N_{replace}$	$N_{pop}/2$
Crossover probability	0.9
Mutation probability	0.2
Generations	100

In case of genetic algorithm for rough set 100 generations were sufficient to obtain reliable results.

3.5 Hybrid rough sets and fuzzy logic algorithm

Multistage classifier created in this thesis can be divided into three phases:

1. Rough sets classifier construction using genetic algorithm presented in section 3.4.2
2. Fuzzy logic classifier construction with rule generation by heuristic approach described in 3.3.4
3. Pattern recognition using sequential hybrid classifier

Each step plays an important role in the whole process and affects the final classification accuracy [13], [29], [34]. Proper parameters of genetic algorithm are especially important. The whole hybrid algorithm can be summarized in the following steps:

1. Divide available dataset into three separated subsets: the first for genetic algorithm operation, the second and third as training and testing.
2. Train rough sets, fuzzy logic classifiers
3. Classify pattern using rough sets algorithm:
    - If pattern is classified by a *certain* or *possible* rule then it is a final label

- If no proper rule is found or more than one rule have the same strength but different label then pattern is rejected and processed by fuzzy logic classifier.

An illustrative scheme of hybrid classifier is presented in fig. 3.4.

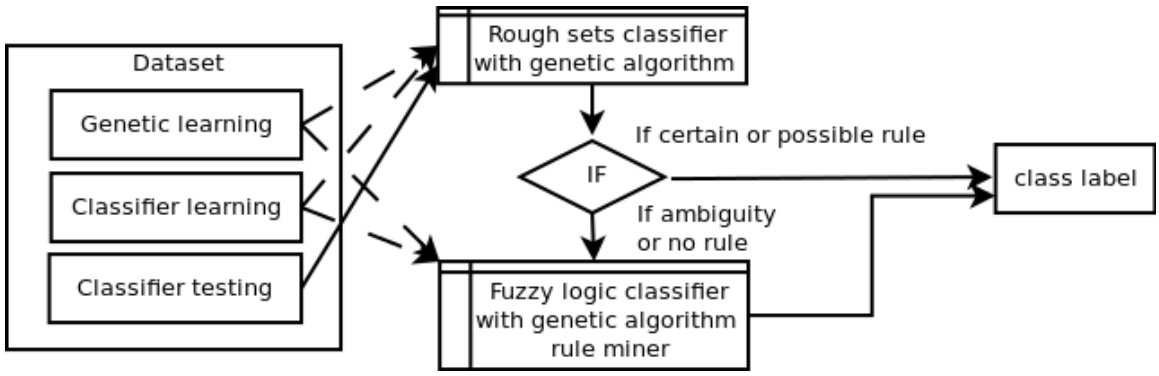

Figure 3.4: Schematic diagram how hybrid classifier works

4 Experimentation system

4.1 Assumptions

This section starts the second part of this thesis. In the first, algorithms were presented in case of construction and their properties. For the simulation purposes the following algorithms have been implemented:

1. Basic Rough sets algorithm
2. Rough sets algorithm with modification of decision rules basing on the granulation step G
3. Fuzzy logic algorithm with genetic approach for rule construction
4. Hybrid algorithm consisting of:
    - Rough sets algorithm with genetic approach for finding the optimal partition in feature space and attribute reduction
    - Fuzzy logic algorithm with genetic approach for constructing decision rules

4.2 Datasets

To perform all the experiments, testing datasets from *UCI* repository were used [49]. This approach is commonly used in the literature because ensures that someone in the future would be able to retake the tests and compare the results. The main motivation in choosing those datasets was to ensure diversity and test algorithms with complicated and complex problems. Below each dataset is shortly described:

- Dataset name: Haberman
    - #attributes: 3
    - #instances: 306
    - #classes: 2
    - Desctiption: This dataset contains cases from a study that was conducted between 1958 and 1970 at the University of Chicago's Billings Hospital on the survival of patients who had undergone surgery for breast cancer.
- Dataset name: Iris

  - #attributes: 4
  - #instances: 150
  - #classes: 3
  - Desctiption: This dataset is one the most commonly used in pattern recognition task. Attribute information:
    - sepal length in cm
    - sepal width in cm
    - petal length in cm
    - petal width in cm

- Dataset name: Wine
  - #attributes: 13
  - #instances: 178
  - #classes: 3
  - Desctiption: These data are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines.

- Dataset name: Thyroid
  - #attributes: 5
  - #instances: 215
  - #classes: 3
  - Desctiption: This dataset was created at the University of California at Irvine by Ross Quinlan during his visit in 1987 for the 1987 Machine Learning Workshop. It contains 5 features describing thyroid symbptoms.

- Dataset name: Bupa
  - #attributes: 6
  - #instances: 345
  - #classes: 2
  - Desctiption: The first 5 variables are all blood tests which are thought to be sensitive to liver disorders that might arise from excessive alcohol consumption. Each line in the bupa.data file constitutes the record of a single male individual.

- Dataset name: Pima
  - #attributes: 8
  - #instances: 768
  - #classes: 2
  - Desctiption: This dataset comes from National Institute of Diabetes and Digestive and Kidney Diseases. The binary-valued decision indicates whether the patient shows signs of diabetes according to World Health Organization criteria. All patients are females at least 21 years old of Pima Indian heritage.
- Dataset name: Wdbc
  - #attributes: 32
  - #instances: 569
  - #classes: 2
  - Desctiption: Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. Each real-valued features are computed for each cell nucleus.

4.3 Efficiency indicators

To evaluate effectiveness of algorithm it has to be consistent approach used in all experiments and additionally a priori knowledge about each dataset must be know. By a-priori knowledge one should understand the label of class for each pattern. Below, there are listed methods of algorithm fitness scoring:

- Classification accuracy CA- the number of correctly classified patterns out of O objects.
  - The best value from n probes

$$B = max\{CA_1, \ldots, CA_n\} \tag{4.1}$$

  - The worst value from n probes

$$W = min\{CA_1, \ldots, CA_n\} \tag{4.2}$$

  - The average value from n probes

$$A = \frac{1}{n} \cdot \sum_{i=1}^{n} (CA_i) \tag{4.3}$$

- Error variance from n simulations

$$\sigma = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(AC_i - A)^2} \tag{4.4}$$

4.4 Program description

To perform all simulations in this project a program was written in *PYTHON*(more information can be found in A). It allows simulating all algorithms with chosen dataset and obtain algorithm efficiency indicators.

Program was tested on Linux platform with Intel Pentium Dual Core 2.4 GHz, 2GB memory. To run the program one has to install *Python* environment at least in version 2.6. Because implemented algorithms have many setting parameters, they were written to the file so that easily change their value in testing procedure. Output results(efficiency indicators) are written to CSV for further processing. The most preferable environment for running this project is Eclipse, free to download from the Internet. Each classifier is written in the form of *Python* class so that further extension would be very easy. An example of classifiers usage is presented in appendix A.

5 Simulation investigations

This section presents environment setup and later the results of simulation investigations. It is important to describe simulation setup so that in the future someone could repeat test or maybe extend the application.

5.1 Simulation environment

When it comes to classification task there is always a problem of how to divide available dataset into training and testing sets. One of the most common approach to ensure proper classifier evaluation is cross validation (more information about cross validation methods can be found in [43]). There are different types:

- **holdout cross validation**- data set is separated into two sets, called the training set and the testing set. Classifier is trained using the training set only. Then the classifier is asked to predict the output values for the data in the testing set. The advantage of this method is that it is usually preferable to the residual method and takes no longer to compute. However, its evaluation can have a high variance, because it depends heavily on which data points end up in the training set and which end up in the test set

- **Leave-one-out cross validation**- the classifier is trained on all the available data except for one point and a prediction is made for that point and stored to compute the average error from all points.
- **K-fold cross validation**- the data set is divided into k subsets, and the hold-out method is repeated k times. Each time, one of the k subsets is used as the test set and the other $k-1$ subsets are merged together to form a training set.

In this thesis 4-fold cross validation was used (see fig. 5.1). To ensure that presented results are reliable each test was repeated 10 times. Additionally, if some patterns had missing attributes in the dataset, they were removed from classification.

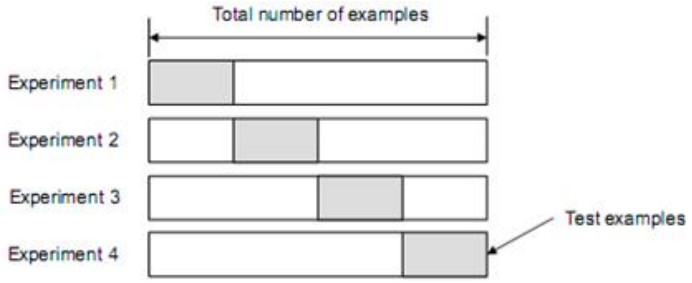

Figure 5.1: Example of 4-fold cross validation.

5.2 Simulation results

5.2.1 Impact of granulation step on rough sets efficiency

The aim of this test was to find out how the discretization of feature space affect the classification accuracy. The number of intervals for each attribute was the same and denoted as K_l, where $l \in (1, \ldots, q)$. Habernam dataset was taken as the input to the system. Dataset was divided into training and testing dataset using 4-fold cross-validation method. The number of objects to recognize was equal to 153. Results of the simulation are presented in table 5.1 where the notation is as follows:

- G- granulation step
- O- total number of correctly recognized objects
- C/CD- number of patterns for which a certain decision rules were used/number of correctly recognized patterns using these rules

- P/PD- number of patterns for which a possible decision rules were activated/number of correctly classified objects using these rules
- V number of patterns rejected from classification. There was no suitable rule or more rules than one have the same strength but different class label.
- C^*, P^*, V^* - total number of *certain*, *possible* or *dummy* (strenght is equal to zero) decision rules, respectively

In the experiment, for every feature the initial step of granulation was changed from 4 to 18 while the factor of its increasing was equal to one.

Analyzing the results of simulation presented in table 5.1 one can see that the quality of the algorithm depends heavily on the step of granulation G and better results are obtained rather for small G. It can be concluded that increasing G results in growing the number of rules with the strength equal to zero (parameter V^*). In the learning phase algorithm is unable to find class representatives so classification is impossible and pattern is rejected. The bigger granulation step G is, more *certain* or *possible* rules are obtained, but on the other hand the number of cells without any representatives is increasing. Generally, it is better to keep V^* parameter rather small and correlate its value with C and P factors. Let notice that for $G = 18$ the classification accuracy is very poor. The number of *certain* rules is the biggest comparing with other cases. The reason for low classification is connected with V^* where 5794 rules are *dummy*.

Table 5.1: Result of simulation for finding the dependency between granulation step and classification accuracy

G	O	C/CD	P/PD	V	C^*	P^*	V^*
4	95	0/0	125/95	28	3	7	54
5	97	0/0	121/97	32	2	4	119
6	52	3/1	65/51	85	5	8	203
7	74	0/0	91/74	62	3	6	334
8	18	2/1	24/17	127	8	9	495
9	52	4/4	59/48	90	6	7	716
10	57	36/25	38/32	79	12	13	975
11	25	5/2	30/23	118	7	9	1315
12	45	49/41	9/4	95	19	12	1697
13	9	0/0	13/9	140	17	12	2168
14	30	9/6	28/24	116	24	14	2706
15	29	30/25	17/4	106	27	13	3335
16	18	2/1	20/17	131	22	12	4062
17	25	3/1	38/24	112	24	17	4872
18	9	2/1	11/8	140	25	13	5794

5.2.2 Impact of recursive modification of granulation step on rough sets algorithm efficiency

In the previous section (5.2.2) it was shown that the granulation step strongly affects the classification accuracy. Greater G implies that we have more *certain* or *possible* rules, but on the other the hand number of patterns without rule covering is increasing. A lot of patterns are rejected because no proper rule is found. To improve that situation an algorithm for modification of decision rules is proposed. More details are presented in section 3.2, but generally when an object is rejected from classification, current G is decreased by $\epsilon = 1$ until *certain* for *possible* rule is found. The same dataset was used as in the previous test with the same algorithm settings to show that this approach is more efficient comparing with the previous algorithm. Results of classification are presented in fig. 5.2 for the basic rough sets (blue line-BRS) and algorithm with modification of decision rules (orange line-MRS).

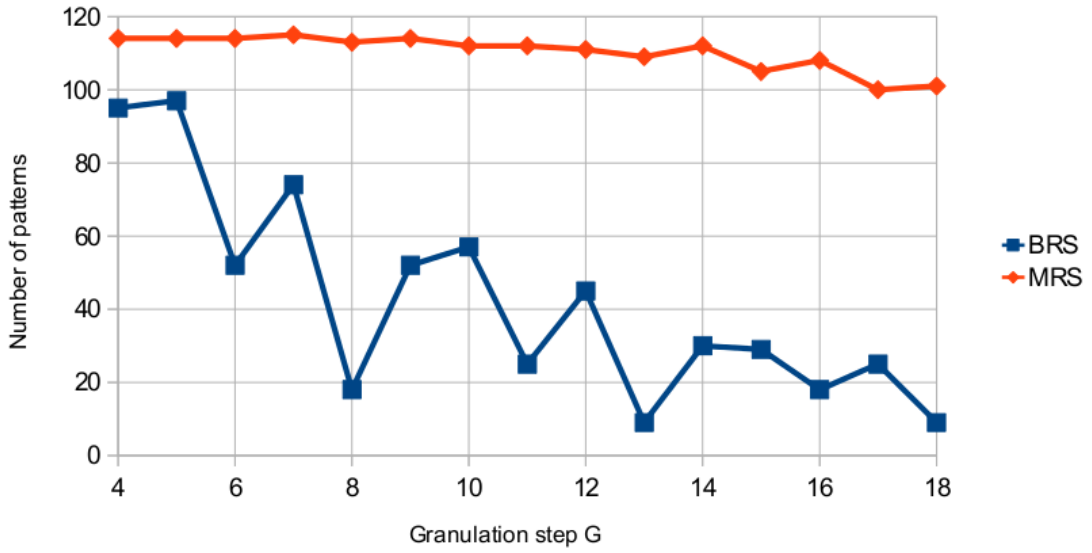

Figure 5.2: Comparison of Basic Rough sets algorithm with algorithm where modification of decision rules is introduced.

Looking at fig. 5.2 one can observe that the modification of granulation step G during algorithm execution increases the classification accuracy and even if algorithm is started with different G the classification stays almost at the same level while in the basic approach the greater G implies worse results. Additionally, it is visible that increasing granulation step is not the right solution. Even if the number of *certain* or *possible* rules is greater, the final result is worse. What is more important, computational time is longer for greater G. In this case the optimal G would be 5, but for each problem G should be chosen independently because it must reflect how patterns are located in the feature space.

The main two disadvantages of proposed rough sets algorithm are as follows:

- It uses an arbitrary chosen step of granulation. Modification of decision rule improves classifier quality, but for the prize of computational time.
- It uses all attributes for creation of decision rules. When the problem is complex then the decision rules are long and tangled. Additionally, some features are useless in classification, instead of valuable information they bring noise to the system and deteriorate final results.

5.2.3 Impact of number of membership functions on genetic fuzzy logic algorithm efficiency

5.2.3.1 Example of rule generation

In this section the results of fuzzy logic classifier simulation are presented. The goal is to show that proposed algorithm construction is correct and gives satisfactory results.

At first, let remind what are the requirements for fuzzy logic classifier in this thesis. For the input it is provided a dataset without no expert knowledge of how to appropriately divide the feature space into fuzzy sets, and how many membership functions are need. The goal is to find minimal rule set with the possible highest classification accuracy.

As the first step let present how fuzzy logic classifier deals with pattern recognition and what is the minimal rule set for exemplary Iris dataset. At the beginning each feature is divided into 14 membership functions in the same way as presented in fig. 3.1 plus one linguistic variable $DON'T\ USE$. The final rule set which was able to classify 32 out of 34 testing patterns is presented in fig. 5.3. From 10 test the following rule construction occurred 7 out of 10 times. In this example each attribute was normalized and after this process feature values were from $< 0, 1 >$ range.

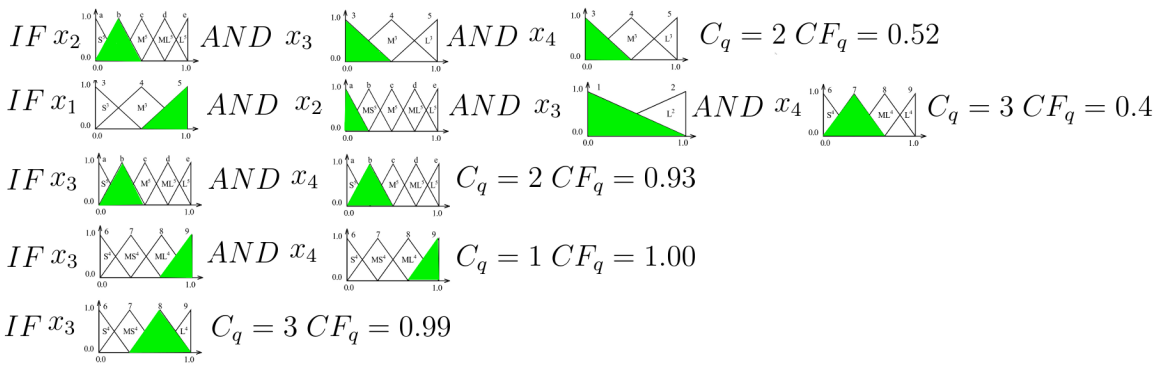

Figure 5.3: Example of rule set generated for Iris dataset

Analyzing figure 5.3 it is visible that some attributes were omitted from classification, but the results of classification are quite satisfactory. Additionally, only five rules were needed for correct pattern recognition.

5.2.3.2 Classification accuracy and the number of membership functions in genetic fuzzy logic algorithm

The goal of the second part of this test was to check how the number of initial membership functions affects the final result of classification. There is a question if it is better to use many small membership functions (for example 14 functions such as presented in fig. 3.1) or only few functions with greater area coverage. In the first case the solution space is much greater than in the second approach so many rules must be created to obtain reliable results. Additionally, in most recognition problems we do not need so precise feature partitions because it can happen that for many created regions we cannot find proper representatives in the training set.

Parameters for genetic algorithms are the same as presented in table 3.1. In each simulation the level of partitions k was changed from 7 to 2. Few words of explanation should be written about how k determines the number of membership functions MF for each attribute. This number is described by eq. (5.1)

$$MF = \sum_{1}^{k}(n+1)+1 \tag{5.1}$$

Wine dataset was used as the input to the system. After dividing the set into three sets(the first one for genetic algorithm, the second and the third for fuzzy logic training and testing) 45 patterns were used for classification.

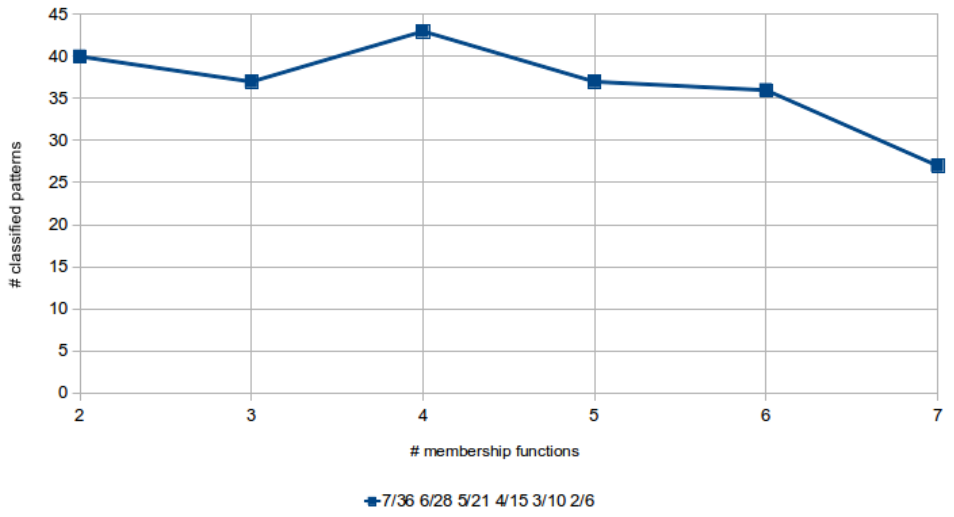

Figure 5.4: Impact of initial number of membership functions MF per attribute on the final fuzzy classification rate

Looking at the fig. 5.4 one can conclude that it is not worth constantly increasing the number of membership functions (MF) per attribute because for greater MF algorithm obtains worse results. From conducted simulations it was concluded that the optimal k value is 4 which leads to 14 different membership functions per attribute. If not implicitly stated, $k = 4$ will be always used in next tests.

5.2.3.3 Classifiers performance comparison

Here the genetic fuzzy logic algorithm is compared with rough sets algorithms. For basic rough sets algorithm granulation step G was set to 4 while for rough sets algorithm with modification of decision rules this factor was equal to 7. Parameters for genetic fuzzy logic classifier were the same as presented in table 3.1, parameter k was set to 4. Basic rough sets algorithm and algorithm with modification of decision rules used all available features for classification. In case of genetic fuzzy logic classifier the average number of attributes used in 10 rules was calculated.

The main purpose of this simulation was to check the efficiency of proposed genetic algorithm, especially how many attributes and how many rules are needed for the proper classification. Table 5.2 presents the results of simulation. Notation used in the table is as follows:

- F- number of attributes in the dataset
- O- number of objects used for classification
- W, B, A, σ-performance indicators described in section 4.3
- RSR- number of objects correctly recognized by basic rough sets algorithm
- $RSMRS$- number of patterns correctly recognized by rough sets algorithm with modification of decision rules
- GFL- number of objects correctly recognized by genetic fuzzy logic classifier
- FU- an average number of features used by genetic rough fuzzy logic classifier in the best rule set

Table 5.2: Comparison of accuracy of classification for genetic fuzzy logic classifier with rough sets algorithms for different datasets

Dataset	F	O	W	B	A	σ	FU
RSR							
haberman	3	77,0	50,0	55,0	53,0	1,9	3,0
iris	4	39,0	27,0	31,0	29,3	1,5	4,0
thyroid	5	56,0	41,0	46,0	44,0	1,9	5,0
bupa	6	87,0	25,0	37,0	30,3	4,3	6,0
pima	8	192,0	85,0	98,0	91,5	4,9	8,0
wine	13	45,0	0,0	2,0	0,8	0,8	13,0
wdbc	30	142,0	10,0	15,0	11,8	2,0	30,0
RSMR							
haberman	3	77,0	53,0	58,0	55,5	2,1	3,0
iris	4	39,0	32,0	38,0	34,8	2,4	4,0
thyroid	5	56,0	48,0	50,0	49,3	0,8	5,0
bupa	6	87,0	23,0	35,0	28,8	4,3	6,0
pima	8	192,0	62,0	69,0	65,5	2,5	8,0
wine	13	44,5	15,0	23,0	19,3	3,0	13,0
wdbc	30	142,3	10,0	15,0	11,8	2,0	30,0
GFL							
haberman	3	77,0	54,0	58,0	55,9	1,2	8,5
iris	4	39,0	29,0	39,0	34,3	3,1	8,2
thyroid	5	56,0	41,0	45,0	43,4	1,2	8,4
bupa	6	87,0	49,0	58,0	53,4	2,7	8,3
pima	8	192,0	124,0	140,0	134,1	4,8	7,7
wine	13	45,0	35,0	42,0	38,0	1,9	8,3
wdbc	30	142,0	107,0	127,0	119,2	5,6	7,9

Results from table 5.2 indicate that genetic fuzzy logic classifier can compete with other classifier. The main advantage is connected with the number of attributes building decision rules. In each case some features are removed from the set.

5.2.4 Impact of granulation step G on genetic rough sets algorithm efficiency

The goal of this test is to check which approach is better:

1. use the same granulation step G for each attribute and additionally take all feature into classification
2. use different partition for each attribute independently and try to remove some features treating them as a noise.

The first approach is simulated by algorithm with modification of decision rules (see section 3.2), while the second is genetic rough sets algorithm (described in section 3.4). Results of simulations are placed in table 5.3. where the notation is as follows:

- F- number of attributes in the dataset
- O- number of objects used for classification
- W, B, A, σ-performance indicators described in section 4.3
- RSR number of objects correctly recognized by basic rough sets algorithm
- $RSMR$ number of patterns correctly recognized by rough sets algorithm with modification of decision rules
- GRR number of objects correctly recognized by genetic rough sets algorithm
- FU number of features used by genetic rough sets algorithm for classification.

Parameters for genetic rough sets algorithm were the same as presented in table 3.2 and for the first rough sets algorithm granulation step G was equal to 4 and for algorithm with modification of decision rules starting granulation value was 7. These parameters were selected from the previous simulations because in such configuration the best results were obtained.

Table 5.3: Accuracy of classification for genetic rough sets and basic rough sets algorithms for different datasets

Dataset	F	O	W	B	A	σ	FU
RSR							
haberman	3	77,0	50,0	55,0	53,0	1,9	3,0
iris	4	39,0	27,0	31,0	29,3	1,5	4,0
thyroid	5	56,0	41,0	46,0	44,0	1,9	5,0
bupa	6	87,0	25,0	37,0	30,3	4,3	6,0
pima	8	192,0	85,0	98,0	91,5	4,9	8,0
wine	13	45,0	0,0	2,0	0,8	0,8	13,0
wdbc	30	142,0	10,0	15,0	11,8	2,0	30,0
RSMR							
haberman	3	77,0	53,0	58,0	55,5	2,1	3,0
iris	4	39,0	32,0	38,0	34,8	2,4	4,0
thyroid	5	56,0	48,0	50,0	49,3	0,8	5,0
bupa	6	87,0	23,0	35,0	28,8	4,3	6,0
pima	8	192,0	62,0	69,0	65,5	2,5	8,0
wine	13	44,5	15,0	23,0	19,3	3,0	13,0
wdbc	30	142,3	10,0	15,0	11,8	2,0	30,0
GRR							
haberman	3	77,0	58,00	61,00	59,50	1,50	2,00
iris	4	39,0	35,00	39,00	37,00	1,58	2,42
thyroid	5	56,0	49,00	54,00	51,17	1,91	2,58
bupa	6	87,0	55,00	61,00	57,67	2,06	2,67
pima	8	192,0	145,00	155,00	149,42	3,15	3,00
wine	13	45,0	40,00	44,00	42,67	1,43	2,92
wdbc	30	142,0	128,00	135,00	132,67	1,65	5,08

From table 5.3 one can conclude that genetic rough sets algorithm obtains better results than other algorithms, especially it is visible for more complex problems such as wine or pima datasets. Additionally, let analyze the last column AU. It determines how many attributes are used in genetic rough sets algorithm for classification. It is noticeable that some features are useless and genetic rough sets algorithm is able to find valuable attributes. Another thing to reconsider is how the complexity of the problem affects algorithm classification accuracy. Basic rough sets algorithm tackles quite well with simple problems, for example iris or haberman datasets, but when the number of attributes is greater than 4 algorithm efficiency decreases, while genetic rough sets is not affected by this problem and can deal with complex datasets.

5.2.5 Comparison of hybrid classifier with other classifiers

In this section results of simulation for hybrid classifier are presented. The accuracy of classification is compared with other classifiers trained and tested with the same datasets. The main goal of this simulation was to check if proposed hybridization (rough set and fuzzy logic) can compete with other classifiers. As the source of reference different types of classifiers were chosen to ensure the greatest diversity:

- LDAC classifier (Lin ear Discriminant classifier)- The linear discriminant analysis method consists of searching some linear combinations of selected variables, which provide the best separation between the considered classes. These different combinations are called discriminant functions (see example in fig. 5.5)

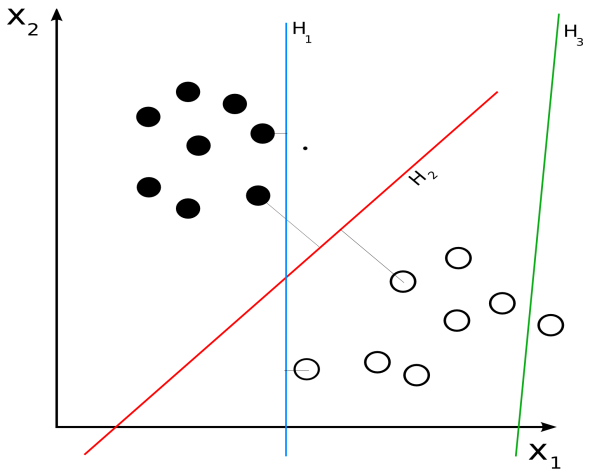

Figure 5.5: Example of linear discriminant classifier for 2-dimensional problem

- 3-KNN Classifier- it is one of the simplest approach in the pattern recognition, it classifies objects based on closest training examples in the feature space.

- Gini index classifier- it is an example of decision tree algorithm where the decision is represented in case of decision rules. Decision node specifies a test on a single attribute, leaf node indicates the value of the target attribute, arc/edge splits of one attribute and indicate the disjunction of test to make the final decision. Decision trees classify instances or examples by starting at the root of the tree and moving through it until a leaf node is reached.

- Maximum likelihood classifier- this classifier is commonly used in image recognition tasks. It assigns a pixel to a class on the basis of its probability of belonging to a class whose mean and covariance are modelled as forming a normal distribution in multidimensional feature space.
- Svm classifier- it is non-probabilistic linear classifier which deals with finding an optimal linear hyperplanes for class separation.

Results of simulation are presented in table 5.4, where notation is as follows:

- O- number of patterns to be recognized
- W, B, A, σ-performance indicators described in section 4.3
- MACL- Maximum Likelihood Classifier
- Hybrid- multistage hybrid rough sets fuzzy logic classifier described in section 3.5

Numbers in bold font indicates this classifier which obtained the best result for a particular dataset. In cases when the same result was obtained for more classifier then the number of attributes is taken into account.

Table 5.4: Comparison of hybrid rough fuzzy classifier with other common classifiers

Classifier	O	W	B	A	σ
iris					
LDAC	39,0	36,0	38,0	36,8	0,8
KNN	39,0	34,0	38,0	36,2	1,8
GINI	39,0	31,0	38,0	35,5	2,9
MACL	39,0	34,0	38,0	36,3	1,5
SVM	39,0	24,0	26,0	25,0	1,0
Hybrid	39,0	35,0	39,0	**37,0**	1,6
Bupa					
LDAC	87,0	56,0	62,0	59,3	2,8
KNN	87,0	55,0	65,0	59,3	4,0
GINI	87,0	54,0	56,0	55,0	0,7
MACL	87,0	48,0	56,0	52,5	3,2
SVM	87,0	53,0	63,0	**59,5**	3,8
Hybrid	87,0	55,0	62,0	58,8	2,6

Continued on next page

Table 5.4 – continued from previous page

Classifier	O	W	B	A	σ
Pima					
LDAC	192,0	146,0	149,0	147,5	1,5
KNN	192,0	131,0	136,0	132,3	2,2
GINI	192,0	123,0	136,0	130,5	5,3
MACL	192,0	142,0	145,0	143,3	1,1
SVM	192,0	128,0	141,0	134,8	4,7
Hybrid	192,0	146,0	152,0	**149,0**	2,1
haberman					
LDAC	77,0	57,0	58,0	57,5	0,5
KNN	77,0	54,0	57,0	55,8	1,1
GINI	77,0	45,0	58,0	52,3	4,7
MACL	77,0	57,0	58,0	57,8	0,4
SVM	77,0	55,0	58,0	56,5	1,1
Hybrid	77,0	58,0	63,0	**60,0**	2,1
wdbc					
LDAC	142,0	133,0	138,0	135,5	1,8
KNN	142,0	127,0	134,0	131,3	2,7
GINI	142,0	128,0	131,0	128,8	1,3
MACL	142,0	133,0	139,0	**135,8**	2,2
SVM	142,0	127,0	135,0	131,8	2,9
Hybrid	142,0	127,0	138,0	134,3	4,3
thyroid					
LDAC	56,0	47,0	51,0	49,3	1,8
KNN	56,0	48,0	52,0	50,8	1,6
GINI	56,0	48,0	53,0	50,5	1,8
MACL	56,0	50,0	54,0	51,0	2,1
SVM	56,0	50,0	53,0	51,0	1,2
Hybrid	56,0	50,0	55,0	**52,0**	2,1
wine					
LDAC	45,0	41,0	43,0	43,0	0,7
KNN	45,0	27,0	35,0	39,9	3,0
GINI	45,0	37,0	41,0	39,0	2,0
MACL	45,0	43,0	45,0	**44,3**	0,8
SVM	45,0	38,0	44,0	41,3	2,4
Hybrid	45,0	41,0	43,0	43,0	0,7

From table 5.4 one can conclude that hybrid classifier obtains quite good results comparing with other classifiers. From seven datasets four times it was the best. In other cases results are comparable. What is more important hybrid classifier is able to classify pattern with reduced number of attributes. In this case created decision rules are simpler and more readable for user. This is especially important in medicine where physician is provided with decision rules and basing on them makes the final diagnosis. Another thing to reconsider is the stability of proposed classifier. It uses genetic algorithm for rule construction, so taking into account its random nature hybrid classifier can be unstable, but simulations show that this is not a problem. Appropriate number of generations for genetic algorithm assures the proper convergence and as the consequence hybrid classifier is stable.

6 Summary and conclusions

6.1 Conclusions from conducted experiments

In this paper the results of simulation investigations were presented. The main purpose of five test scenarios was to evaluate prepared classifiers in case of classification accuracy. Implemented and tested algorithms in this paper are as follows:

1. Basic Rough sets algorithm
2. Rough sets algorithm with modification of decision rules
3. Genetic Based Fuzzy Logic algorithm
4. Multistage hybrid classifier using Rough sets and Fuzzy logic

Author intention was to present the whole process of constructing complex classifier from simpler ones. Researches started with the basic rough sets algorithm. Because the results of classification were not satisfactory new algorithm with modification of decision rules was introduced. Significant improvement was visible for simple problems, but for multidimensional datasets the classification accuracy remained poor. The next proposal consisted in constructing multistage hybrid classifier where the power of fuzzy logic and rough sets reasoning were connected. Original algorithm presented in this thesis used genetic algorithm for finding the reduct of attributes and the optimal granulation step G for each attribute in case of rough sets and for fuzzy logic genetic algorithm determined the best decision rule set. Conducted test confirmed that proposed algorithm can compete with other classifiers and obtains reliable results. Next paragraphs will shortly summarize each research.

The main goal of the first simulation was to check how the granulation step G affects the classification accuracy. The efficiency of rough sets algorithm is determined by the number of *certain* and *possible* rules. Ideally, it would be great that all rules are *certain* and we have 100% coverage in the rule set. Conveyed simulations confirmed the thesis that granulation step has the greatest impact on the classification results. This parameter must be chosen very carefully and few factors must be taken into account:

- the complexity of the problem, how many attributes are used to describe the pattern
- how the granulation intervals are generated, equally for all attributes or independently

Another important conclusion from the first test concerns the optimal value for G. It is not worth increasing G, good results of classification are obtained rather for small numbers such as five or six. As the evidence analyze table 5.1 where for $G = 18$ the classification accuracy was very poor. Increasing G ensures that the number of *certain* decision rules is greater, but on the other hand there are more dummy rules (rule with strength equal to 0).

Taking into account the fact that the classification accuracy of algorithm presented in section was not satisfactory, the recursive rough sets algorithm was proposed to tackle with situation where for the particular granulation G there is no appropriate rule in the rule sets. From simulations it can be concluded that this approach improved the classification significantly and what is more important, thanks to the modification of decision rules algorithm was not affected by the granulation step G. The stability of classifier is the biggest advantage, but on the other hand the main drawback is connected with algorithm execution. In situation when the pattern is rejected from classification algorithm is recursively invoked until a proper rule is found. This requires much more time than the basic approach without rule modification. For simple classification tasks this is not a problem, but for more complex datasets algorithm computation is much longer.

This thesis concerns the problem of classifying unknown patterns using rough sets approach and fuzzy logic reasoning. To successfully accomplish this task new algorithm of fuzzy logic had to be implemented. In the literature there are many examples of fuzzy logic controllers or classifiers. In this paper genetic based fuzzy logic classifier was implemented. The main advantages of proposed solution are as follows:

- it manages to construct decision rules without any expert knowledge. It means that the input to the system constitutes only raw data from dataset without additional information about the number of membership functions or their location in the feature space. As the output we obtain rule set with the highest classification rate.

- it is able to apply attribute reduction

In section it was shown that proposed fuzzy classifier was able to classify 32 out of 34 testing patterns from iris dataset with five rules (presented in fig. 5.3).

As fuzzy logic was finished and genetic algorithm turned out to be a good solution it was decided to connect genetic algorithm with rough sets and check how this fusion works. The main goal of the heuristic approach was to find an optimal granulation for each feature independently and try to remove those attributes that are useless in classification. Conducted tests proved that this direction of researches is good and gives promising results. Especially it was visible for high

dimensional datasets where the classification rate for basic rough sets and rough sets with modification of decision rules was very low.

The last part in this master thesis consisted in constructing a multistage hybrid classifier connecting implemented by the author genetic rough sets algorithm and genetic based fuzzy logic algorithm. There are many types and methods of classifier fusion, but in this paper two-phased sequential classification was proposed. Comparison with different well-known classifiers gave optimistic results, but of course more profound test are required to fully prove its usefulness.

6.2 General conclusions

To sum up, this paper presents the result of work in the field of pattern recognition and classification. It reviewed some of the most representative publications connected with rough sets, fuzzy logic and genetic algorithms. Here an original construction of hybrid classifier has been presented. Attached results are very promising and encourage for further more complex investigations. Generally, pattern recognition is not an easy task and many factors and parameters must be taken into account. Until now, no-one has managed to invent the classifier with 100% accuracy. Even if for one dataset results are satisfactory, but for others classification will be worse.

7 Future work

In this thesis the basic parameters and settings were checked for rough sets algorithm and fuzzy logic algorithm. In the future, it is strongly recommended to carry out more profound simulations to fully understand behaviour of algorithm in different environment and settings. In this paper to evaluate each classifier well-known datasets from *UCI* Repository were used. This offers reliable environment for testing, but the next step in the future work is to apply proposed algorithm into real life problem, such as optimal control or image pattern recognition. Using rough sets properties it would be advisable to detect tumor tissue on CT, MRI images or bone structures for further 3D reconstruction.

Another thing to reconsider in the next researches is how to generate partition of feature. Here, the genetic algorithm was used to find the reduct of attributes and the number of intervals for each attribute independently. Results of simulations confirmed the usefulness of this approach, but here arises the question if there is another solution for finding an optimal feature granulation. Two possible future tests:

1. Testing the classification accuracy of rough sets algorithm when granulation is based on the frequency of patterns in the training dataset. This approach assumes that for clusters with many patterns the granulation will be more precise while in other places it would be sparse.

2. Testing the classification accuracy of rough sets algorithm when granulation is determined by fuzzy logic and triangular membership functions. A concept of fuzzy discretization of feature space for a rough sets theoretic classifier is presented in [2]

The last, but not least aspect of the future work is to check different types of classifier hybridization. In this thesis rough sets algorithm was the most important and only in cases when pattern was rejected fuzzy logic classifier was used. It would be required to simulate different scenarios of classifier ensemble, for example majority voting with fuzzy logic, rough sets and neural network or 3-KNN classifiers. Additionally, it would be great to compare different algorithms for feature reduction with genetic approach used in this paper.

References

[1] Negnevistky M., : *"Artificial Intelligence: A Guide to Intelligent Systems"*, Second Edition, Addidon Wesley, November 12, 2004

[2] Roy A., Pal K. S., : *"Fuzzy discretization of feature space for rough set classifier"*, Elsevier, Pattern Recognition Letters 24, 2004

[3] Krupka I., JIRAVA P., : *"Modelling of Rough-Fuzzy Classifier"*, WSEAS TRANSACTIONS on SYSTEMS, Issue 3, Volume 7, March 2008

[4] Khoo L.P., Zhai L., : *"A prototype genetic algorithm-enhanced rough set-based rule induction system"*, Elsevier, Computers in Industry 46, 95-106, 2001

[5] Meng D., Pei Z. , : *"Extracting linguistic rules from data sets using fuzzy logic and genetic algorithms"*, Elsevier, Neurocomputing 78, 48-54, 2012

[6] Kothari A., Keskar A., : *"Feature Space Reductions Using Rough Sets for a Rough-Neuro Hybrid Approach Based Pattern Classifier"*, Proceedings of the World Congress on Engineering and Computer Science 2008 WCECS 2008, October 22 - 24, 2008, San Francisco, USA

[7] Hu Q., Shuang A., : *"Robust fuzzy rough classifiers"*, Elsevier, Fuzzy Sets and Systems 183, 26–43, 2011

[8] Wu W., Peirong L., : *"Topological Spaces for Fuzzy Rough Sets Determined by Fuzzy Implication Operators"*, Sixth International Conference on Fuzzy Systems and Knowledge Discovery, 2009

[9] Choudhari A., Nandi G.C., : *"NRC: A Neuro-Rough Classifier for Landmine Detection"*, IEEE lndicon 2005 Conference, Chennai, India, 11-13 Dec. 2005

[10] Affanso C., Sassi R. J., : *"Traffic Flow Breakdown Prediction using Feature Reduction through Rough-Neuro Fuzzy Networks"*, Proceedings of International Joint Conference on Neural Networks, San Jose, California, USA, July 31 – August 5, 2011

[11] Wang X., Hua Z., : *"A Hybrid Text Classification model based on Rough Sets and Genetic Algorithms"*, Ninth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2008

[12] Qinghua H., Congxin W., : *"Fuzzy preference relation rough sets"*, Harbin Institute a/Technology, Harbin 150001, P. R. China, 2007

[13] Mitra S., Banka H., : *"Rough–Fuzzy Collaborative Clustering"*, IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART B: CYBERNETICS, VOL. 36, NO. 4, AUGUST 2006

[14] Feng L., Liu Z., : *"Genetic Algorithms and Rough Fuzzy Neural Network-based Hybrid Approach for Short-term Load Forecasting"*, IEEE, 1-4244-0493-2, 2006

[15] Wu Q., Wang T., Ji-Sheng L., : *"NEW RESEARCH ON FUZZY ROUGH SETS"*, Proceedings of the Fifth International Conference on Machine Learning and Cybernetics, Dalian, 13-16 August 2006

[16] BING-ZHEN S., ZENG-TAI G., : *"THE FUZZY DESCRIPTION OF THE BOUNDARY REGION IN ROUGH SETS AND ITS APPLICATIONS"*, Proceedings of the Sixth International Conference on Machine Learning and Cybernetics, Hong Kong, 19-22 August 2007

[17] MENG-XIN L., CHENG-DONG W., : *"A VISION-BASED INSPECTION SYSTEM USING FUZZY ROUGH NEURAL NETWORK METHOD"*, Proceedings of the Fifth International Conference on Machine Learning and Cybernetics, Dalian, 13-16 August 2006

[18] Wei-Zhi W., Ju-Sheng M., : *"Generalized fuzzy rough sets"*, Elsevier, Information Sciences 151, 263–282, 2003

[19] Shen Q., Chouchoulas A., : *"A rough-fuzzy approach for generating classifcation rules"*, Pergamon, Pattern Recognition 35, 2425 – 2438, 2003

[20] Qinghua H., Zongxia X., : *"Hybrid attribute reduction based on a novel fuzzy-rough model and information granulation"*, Elsevier, Pattern Recognition 40, 3509 – 3521, 2007

[21] Surmann H., Selenschtschikow A., : *"Automatic generation of fuzzy logic rule bases: Examples I"*, PROC. OF THE NF2002: FIRST INTERNATIONAL ICSC CONFERENCE ON NEURO-FUZZY TECHNOLOGIES, PP 75, CUBA 16-19 JAN. 2002

[22] Surmann H., Maniadakis M., : *"Learning feed-forward and recurrent fuzzy systems: A genetic approach"*, Elsevier, Journal of Systems Architecture 47, 649-662, 2001

[23] Ishibuchi H., Yamamoto M., : *"Fuzzy Rule Selection by Multi-Objective Genetic Local Search Algorithms and Rule Evaluation Measures in Data Mining "*, Department of Industrial Engineering, Osaka Prefecture University, 2003

[24] Wen-June W., Tzu-Guan Y., : *"A method of Self-Generating Fuzzy Rule Base via Genetic Algorithm"*, Department of Electrical Engineering, National Central University, 2006

[25] Vassilis S.K., Petrounias I., : *"Intelligent Classification using Adaptive Fuzzy Logic Systems"*, 4th International IEEE Conference "Intelligent Systems", 2008

[26] Zadeh L.A., : *"Granular Computing-Computing with Uncertain, Imprecise and Partially True Data"*, Department of EECS, University of California, Berkeley, USA, 2000

[27] Mendes R., Voznika A., Nievola J.C., : *"Discovering Fuzzy Classification Rules with Genetic Programming and Co-evolution"*, PUC-PR, PPGIA-CCET, Springer-Verlag Berlin Heidelberg 2001

[28] Hossein R., Ellis T., : *"A Genetic Fuzzy Approach for Rule Extraction for Rule-Based Classification with Application to Medical Diagnosis"*, Quantitative Imaging lab, Faculty of Computing, Information Systems and Mathematics, Kingston University, 2001

[29] Chiu L. S., : *"An Efficient Method for Extracting Fuzzy Classification Rules from High Dimensional Data"*, Advanced Computational Intelligence, Vol. 1, No. 1, 1997

[30] Abadeh M. S.,: *"Induction of Fuzzy Classification systems via evolutionary Aco-Based algorithms"*, omputer Engineering Department, Sharif University of Technology, Tehran, Iran, IJSSST, Vol. 9, No. 3, September 2008

[31] Pelekis N., Kopanakis I., : *"Fuzzy Miner-A Fuzzy System for Solving Pattern Classification Problems"*, Department of Informatics University of Piraeus, Athens, Hellas, 2001

[32] Ishibuchi H., Yamamoto M., : *"Hybridization of Fuzzy GBML Approaches for Pattern Classification Problems"*, IEEE Trans. on Systems, Man, and Cybernetics-Part B: Cybernetics, vol. 35, no. 2, pp. 359- 365, April 2005

[33] Ishibuchi H., Yamamoto M., : *"Fuzzy Rule Selection by Data Mining Criteria and Genetic Algorithms"*, Department of Industrial Engineering, Osaka Prefecture University, 1998

[34] Gonzales A., Herrera F., : *"Multi-stage Genetic Fuzzy Systems Based on the Iterative Rule Learning Approach"*, Mathware & Soft Computing 4, 233-249, 1997

[35] Dehzangi O., Zolghadri M., : *"Efficient fuzzy rule generation: A new approach using data mining principles and rule weighting"*, Fouth International Conference of Fuzzy Systems and Knowledge Discovery, 2007

[36] Kurzynski M., Zolnierek A., : *"Rough sets and fuzzy sets theory applied to the sequential classification- algorithms and applications"*, Polish Journal of Enviromental Studies, Vol. 17, No. 2B, pp. 68-77, 2008

[37] Zolnierek A., Majak M., : *"Rough sets approach to the problem of classification"*, Proceedings of International Conference MOSIS'X, Czech Republic, pp. 109-114, 2010

[38] Zolnierek A., Majak M., : *"Rough Sets Approach to the Classification Task with Modification of Decision Rules"*, 11th WSEAS INternational Conference on Systems theory and Scientific Computation, Folerence 2011

[39] Grzymala-Busse J., : *"A system for learning from examples based on rough sets"*, Intelligent decision support: handbook of applications and advances of the rough sets theory, R. Slowinski, Dodrecht, Kluver Academic Publishers, Vol.X, No.X, pp. 3-18, 1992

[40] Pawlak Z., : *"Rough sets, decision algorithms and Bayes' theorem"*, European Journal of Operational Research, Vol. 136, pp. 181-189, 2002

[41] Stefanowski J., : *"On rough set based approaches to induction of decision rules"*, Rough Sets in Knowledge Discovery, Physica-Verlag, Heidelberg, New York, Part 1, pp.500-529, 1998

[42] Pawlak Z., : *"Rough Sets"*, Institute of Theoretical and Applied Informatics, Polish Academy of Sciences, 1998

[43] PAYAM R., LEI T., HUAN L., : *"Cross-Validation"*, Arizona State University, 2008

[44] Xia S. , Jamshidi M., : *"A Genetic Algorithms - Discrete Event Simulation Methodology for Modeling and Simulation of Autonomous Systems"*, Department of Electrical and Computer Engineering and Autonomous Control Engineering (ACE) Center, University of New Mexico, Albuquerque, NM 87131, 1999

[45] Nazan K., : *"Population Sizing in Genetic and Evolutionary Algorithms"*, Illinois Genetic Algorithms Laboratory Department of General Engineering University of Illinois at Urbana-Champaign, 2003

[46] Harik G., Cantu-Paz E., Goldberg D. E, Miller B. L., : *"The Gambler's Ruin Problem, Genetic Algorithms, and the Sizing of Populations"*, Illinois Genetic Algorithms Laboratory University of Illinois Urbana, IL 61801 USA, 1999

[47] Chen J., : *"Theoretical Analysis of Multi-Objective Genetic Algorithms - Convergence Time, Population Sizing, and Disequilibrium"*, Department of Information Engineering and Computer Science Feng Chia University, Taichung, Taiwan 407, ROC, 2000

[48] Guney K., Sarikaya N., : *"Comparison of Mamdani and Sugeno Fuzzy inference system models for resonant frequency calculation of rectangular microstrip antennas"*, Progress In Electromagnetics Research B, Vol. 12, 81–104, 2009

[49] www.ics.uci.edu/ mlearn/MLRepository.html

[50] Swiniarski R.W., : *"Rough Sets methods int feature reduction and classification"*, Int. J. Appl. Math. Comput. Sci., Vol. 11, No. 3, 565-582, 2001

List of Figures

List of Tables

Appendix A Program description

A.1 Installation requirements

For the test purposes the simulator in *Python* language was written. To successfully ran this program few requirements must be met. In this master thesis this software was executed on Linux platform and here this approach will be described. Of course it is possible to run the program on Windows, but proper preparation must be undertaken. Requirement for the software:

- *Python* in version at least 2.6
- NumPy
- SciPy
- mlpy library with Gsl. Steps for the proper installation:
  1. sudo apt-get install python2.6-dev
  2. go to: *http://www.gnu.org/prep/ftp.html*
  3. click on an ftp link close to your location
  4. find the gsl/ directory and click on it
  5. find the gsl-VERSION.tar.gz file, where version is 1.14 or greater. Click on that file to download it.
  6. In a terminal window extract the tar.gz file using tar -xzf gsl-VERSION.tar.gz and then cd to the ./gsl-VERSION directory
  7. Look at the INSTALL file. It will probably tell you to run ./configure, then make, and then make install
  8. download mlpy from *http://sourceforge.net/projects/mlpy/files/*
  9. unzip file and inside directory run from command line python setup.py install

The whole project is divided into modules and each classifier is implemented as python class:

- BasicClassifiers- this class implements basic classifiers such as: LDAC, 3-KNN, MaximumLikelyHood Classifier, Gini Index Classifier, svm Classifier
- RoughSetsClassifier- this class implements basic rough sets classifier. Depending on the chosen module it is an algorithm with modification of decision rules or not.

- GeneticFuzzyLogicClassifier- this class simulate genetic fuzzy logic classifier. In the beginning genetic algorithm is run to obtain the best decision rule set and later classification is done
- GeneticRoughSetsClassifier- this class implements genetic rough sets classifier. It comprises of two parts:
  - genetic algorithm for obtaining an optimal partition for each feature
  - classification procedure which uses partition from the previous step for pattern recognition
- HybridClassifier- this class implements hybrid classifier. This is a multistage classifier in which rough sets algorithm is treated as the first classifier and fuzzy logic as the second.

A.2 Example usage

To run each classifier few basic steps must be done. First of all proper parameters with cross-validation and dataset type must be chosen. Below, a simple example is presented showing how to run genetic rough sets classifier fir iris dataset

```
1: #!/usr/bin/python
2: # -*- coding: utf-8 -*-
3:
4: from genetic_rough_sets_classifier import GeneticRoughSetsClassifier
5:
6: if __name__ == '__main__':
7:     K_FOLD_NUMBER = 4
8:     K = 1
9:     GENERATIONS = 500
10:    MUTATION = 0.3
11:    CROSS_OVER = 0.9
12:    G = 7
13:    POPULATION = 10
14:    grs = GeneticRoughSetsClassifier(debug=DEBUG)
15:    filename = 'datasets/iris.data.txt'
16:    grs.read_data(filepath=filename, label_is_last=False)
17:    grs.prepare_data(k_fold_number=K_FOLD_NUMBER)
18:    grs.k_fold_cross_validation(k=K)
19:    grs.initialize_genetic(generations=GENERATIONS, mutation_prop=MUTATION, ↩
           crossover_prop=CROSS_OVER)
20:    grs.create_population(population_size=POPULATION, division=G)
21:    # run genetic algorithm
22:    grs.run()
23:    # genetic algorithm found granulation
24:    # now classify patterns
25:    classified = 0
26:    for i in range(len(patterns)):
27:        if labels[i] == grs.classify(patterns[i]):
28:            classified += 1
29:    grs.print_summary()
```

A.3 Results

In each simulation results of classification for each classifier are saved in csv file. Names of the files are as follows:

- *results/rough_sets_classifier.csv* for RoughSetsClassifier
- *results/genetic_fuzzy_logic_classifier.csv* for GeneticFuzzyLogicClassifier
- *results/genetic_rough_sets_classifier.csv* for GeneticRoughSetsClassifier
- *results/hybrid_classifier.csv* for hybrid classifier

After execution of the program each file is created from scratch so be sure that the previous version of results is save in another location or another name.