POLITECHNIKA WROCŁAWSKA
WYDZIAŁ ELEKTRONIKI

KIERUNEK: INFORMATYKA (ANG.)

PRACA MAGISTERSKA

Badanie efektywności równoległej realizacji algorytmów genetycznych i programów ewolucyjnych

Efficiency evaluation of parallel genetic algorithms and evolutionary programs

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

Efficiency evaluation of parallel genetic algorithms and evolutionary programs

Badanie efektywności równoległej realizacji algorytmów genetycznych i programów ewolucyjnych

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 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. It can 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 **??**.

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 and fuzzy logic algorithms are used and investigated. General purpose of this thesis is to propose hybrid classifier using the power of fuzzy logic and rough sets. First of all the mathematical background 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 [1], [2]. 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 constructing advanced hybrid classifier from the basic algorithms. At first, the basic properties of rough sets algorithm is checked in simulations, later the algorithm for modification decision rules is introduced and at the end as the final point hybridization of algorithms is proposed.

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 **??**. 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.

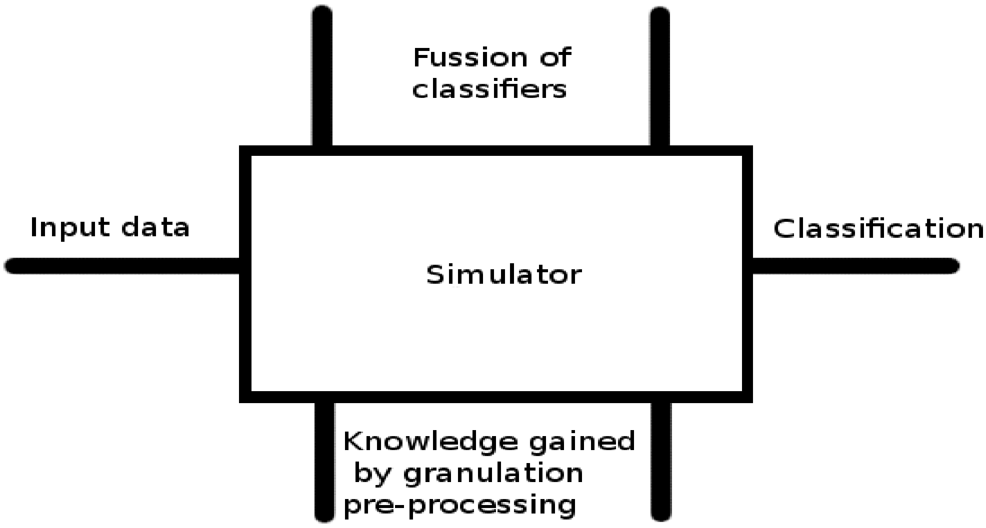


Figure 1.1: General schema of simulation environment. Input/output of the system

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 $C++$ to simulate standard sequential genetic algorithm and its parallel counterparts. Results with comments are placed in section **??**. Plans for the future work and general conclusion are placed in sections 7, 6 respectively.

2 Pattern recognition algorithms

2.1 Introduction

This section is devoted for presenting information about pattern recognition algorithms. The main purpose it 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:

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. When we take into account applied approach pattern recognition, we can distinguish syntactic and statistical pattern recognition. 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 subpatterns. 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

In this section the problem statement will be presented in case of pattern recognition. For the purpose of this thesis we assume 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. Each object will be described by a d-dimensional measured feature vector $x \in X$. In order to classify unknown pattern we use knowledge stored in training set consisting of N training patterns

$$S = (x_1, j_1), (x_2, j_2), \ldots, (x_N, j_N)$$

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

$$i = \Psi(S, x),\ i \in M$$

In decision theory, to ensure that Ψ approximate the problem as closely as possible an additional loss function is introduced that assigns a specific value to 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 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 mathematical approach to deal with imperfect knowledge. In the standard approach we need precise information about pattern to recognize, while rough sets can deal with vague or incomplete data. The problem of imperfect information was tackled for a long time and it became 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 that it works only on the raw data, no additional information are needed such as density probability in Bayesian algorithm. The main facts about rough sets can be summarized in few point presented below:

1. provides attribute reduction
2. generates set of easy to understand and readable decision *IF-THEN* rules
3. evaluate 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.

The nature of problem we met is much more complicated than simple decision that 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*. It can be described by 4-tuple

$$IS =< U, Q, V, f >$$

- U is the universe of discourse which is a finite set of objects
- Q is a finite set of attribute by which each patterns manifests itself

- $V = \bigcup V_q$, 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. Now, 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 as follows

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

Now having I_p we can say that objects x and y are P-indiscernible by a set of attributes P is $y \, I_p \, x$. 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:

$$I_p = \{y \in U : y \, I_p \, x\}$$

By representing a target concept X as a subset of U we would like to describe it with respect to R. Additionally let introduce P as non-empty subset of attributes from Q. In rough sets reasoning 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:

$$\underline{I_p} = \{x \in U : I_p(x) \subseteq X\}$$

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:

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

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.

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. Using above notions we can define P-boundary region which is a difference between upper and lower approximation.

$$BN_p(X) = \overline{I_p} - \underline{I_p}$$

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 it is crisp, otherwise we deal with inexact set which is called rough set. Until that 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.

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

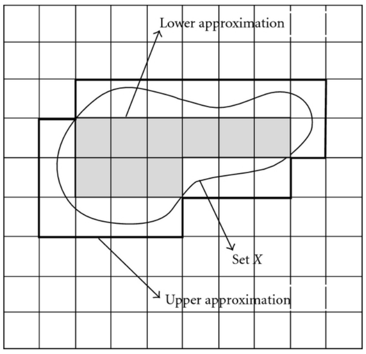


Figure 2.1: Rough sets example.

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:

$$\alpha_p(X) = \frac{\underline{I_p}(X)}{\overline{I_p}(X)}$$

The quality of approximation of $X \subseteq U$ by attributes from subset P represents the percentage of correctly classified objects using attributes P from subset X

$$\gamma_P(X) = \frac{\overline{I_p}(X)}{|X|}$$

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

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

These indicators can be used in determining the quality of rough set algorithm or for finding the optimal reduct.

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.

In rough sets theory we can define four types of data vagueness

- $\overline{I_p}(X) \neq \emptyset$, $\underline{I_p}(X) \neq \emptyset\ IFX$ 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
- $\overline{I_p}(X) \neq \emptyset$, $\underline{I_p}(X) \neq U\ IFX$ 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
- $\overline{I_p}(X) \neq \emptyset$, $\overline{I_p}(X) = U\ IFX$ 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
- $\overline{I_p}(X) \neq \emptyset$, $\overline{I_p}(X) = U\ IFX$ 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 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.

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:

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

2.3.6 Rough sets reasoning from data

The category description can be done in two ways:

1. extensional
2. intentional

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. We will associate decision rules with decision table T.

In this section a practical example will be presented to clear all the things out. As an example let reconsider well-known example of patients suffering from flu.

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

Table 2.1 represents an information system about healthy patient and those suffering from flu. Attributes: Headache, Muscle-pain, Temperature are called condition attributes ($c_i\, for\, i\, in\, (1, \ldots, q)$), while the attribute "Flu" (last column in table 2.1) is considered as decision attribute c_d. Each row of a decision table determines a decision rule. All rules are presented below:

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'

Here we can generate few 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. Using Headache and Muscle-Pain we can divide the set 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: "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 tell us if the set of attributes to describe 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.

2.4 Fuzzy logic

2.4.1 Introduction

In fuzzy logic an element membership to a set is described by membership function which assigns value from interval $< 0, 1 >$. It is a superset of Boolean logic that has been extended to handle the concept of partial truth - values between "completely true" and "completely false". It was introduced by Dr. Lotfi Zadeh in the 1960's as a tool to model the uncertainty of natural language. As in the section 2.3.1, let introduce the problem of defining if person is small or high.

First of all, we have to define a fuzzy set *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". To each person in the universe of discourse, we have to assign a degree of membership in the fuzzy subset *TALL*. The easiest way to do this is with a membership function based on the person's height. Given this definition, 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 common met in the literature are: triangular, trapezoidal, Gaussian shapes (see example in fig. 2.2).

2.4.2 Fuzzy reasoning from data

Reasoning in fuzzy logic is based on decision rules the same as in rough sets approach. Rules are expressed in the form of IF *COND* THEN *DECISION* which can be divided into antecedent set and one consequent determining the output of the rule.

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. Fuzzy logic classification is based on three steps:

1. Fuzzyfication- in the fuzzyfication process we convert continuous quantity into fuzzy number. 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 Mandami inference system.

3. Deffuzification- the process of producing a quantifiable result in fuzzy logic, given fuzzy sets and corresponding membership degrees. These will have a number of rules that transform a number of variables into a fuzzy result, that is, the result is described in terms of membership in fuzzy sets.

Let reconsider a simple example, which should clear all ambiguities (the following example is based on [**?**]). 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. 2.2)

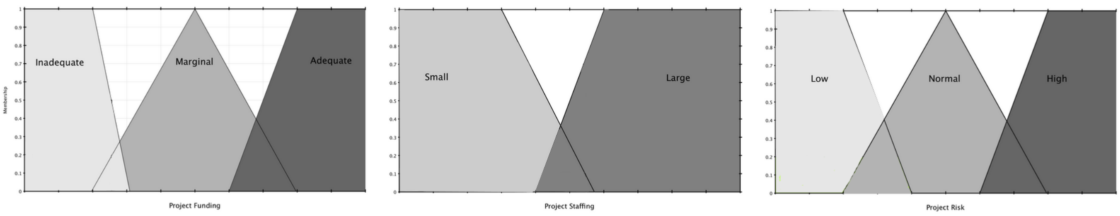


Figure 2.2: Input and output fuzzy linguistic variables for the described system.

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 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.

$\mu_{inadequate}(35) = 0.5$
$\mu_{marginal}(35) = 0.2$
$\mu_{adequate}(35) = 0.0$
$\mu_{small}(60) = 0.1$
$\mu_{large}(60) = 0.7$

2. Now it is time for 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. 2.3), 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. (2.1)

$$CO = \frac{\sum_{x=a}^{b} \mu_A(\chi) \cdot x}{\sum_{x=a}^{b} \mu_A(\chi)} \tag{2.1}$$

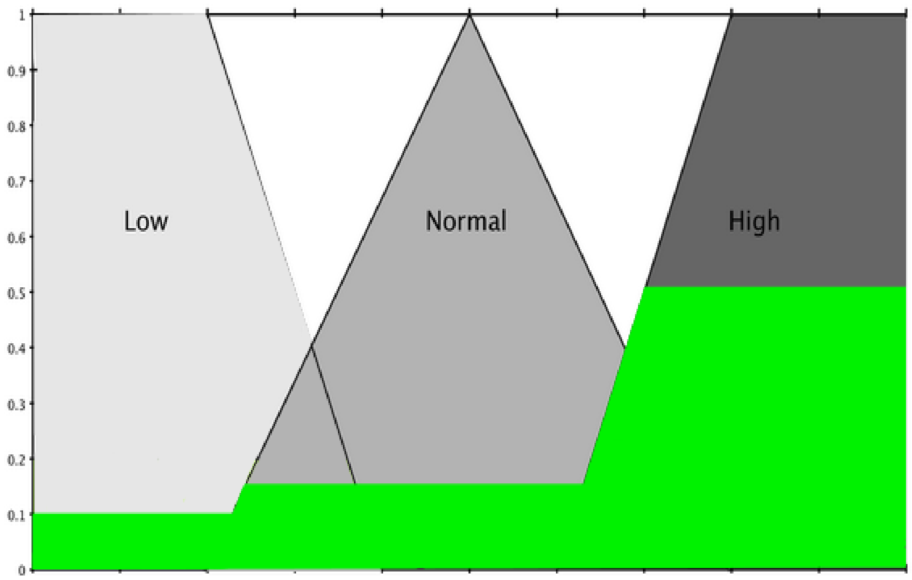


Figure 2.3: Example of clipping the consequent membership function as the result of rule induction

Using eq. (2.1) the output of the system is as follows:

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

2.4.3 Genetic-based machine learning approaches

In the literature one can find many examples of fuzzy logic applications. 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 available so some techniques of data mining must be applied to extract rules from training set.

At this point it must be strongly emphasize that the main focus is based on fuzzy algorithm for pattern recognition task where there is no prior knowledge about fuzzy system. There are different approached from optimization techniques such as gradient descent to heuristic. In this thesis the genetic algorithm will be used.

If until this point something is unclear reconsider the following example. What we have is the problem dimensionality and the set of training patterns. Now the questions arise such as: how to divide feature space into fuzzy set, how generate rule. For example, we create 10 triangular membership functions for each attribute, but is this number optimal. Secondly, when we have 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 Sonar dataset from *UCI* repository has 60 attributes) it is impossible to find the proper combination in a reasonable time.

There are two main methods of genetic-based machine learning approaches:

1. Michigan template- it is a population of fuzzy rules and a single fuzzy rule is handled as an individual. 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
2. Pittsburgh template- in this approach a set of fuzzy rules is handled as an individual. In this case the length of a single individual is equal to $n \cdot N_{rule}$, where n is the length of a 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

2.5 Genetic algorithm

Genetic Algorithm is an element of evolutionary computation, which is a rapidly growing area of soft computing. 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 2.4):

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).

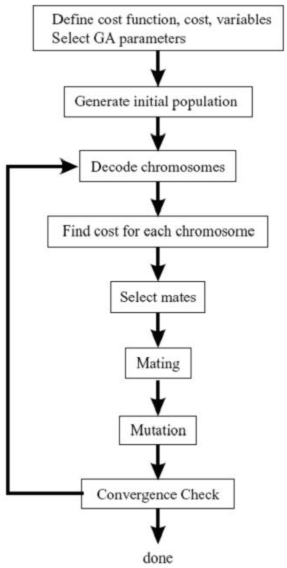


Figure 2.4: Diagram representing phases in genetic algorithm evaluation

Genetic algorithms are widely used as a search techniques in the various fields. In this thesis it will be used for finding 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.

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 .

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.

2.6 Hybrid classifiers

In the recent years, there is an increasing interest in methods of combining multiple learning systems into hybrid one. 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. 2.5:

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. In this thesis the second approach is implemented.

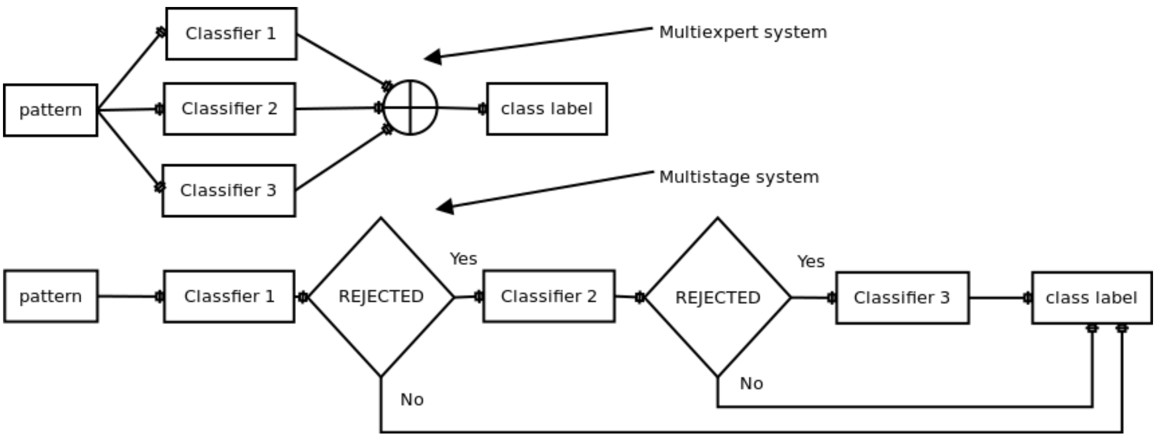


Figure 2.5: Example of two approaches for constructing hybrid classifiers

3 Algorithm construction

In this section algorithm construction will be presented. In this thesis four algorithms were used for pattern recognition task. Here we start with the basic algorithm of rough sets, later present rough sets algorithm with modification step and at last multistage hybrid algorithm consisting of genetic algorithm, rough sets and fuzzy logic will be 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:

1. If the attributes are the real numbers then the granulation preprocessing is needed first. 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 $v^l_{p_l}$ define its p_l interval from $p_l = (1, \ldots, K_l)$

2. Using training dataset construct set $FOR(C)$ of decision rules of the following form:

   $$IF\,(x^1 = v^1_{p_1})\,AND\,(x^2 = v^2_{p_2})\,AND\,\ldots\,(x^q = v^q_{p_q})\,THEN\,\Psi(S, x) = j$$

   Each generated rule is evaluated and strength factor is assigned accuracy of approximation (see section)

3. For the created set of formulas $FOR(C)$ for each $j = 1, \ldots, m$ we calculate lower, upper approximation and the boundary region.

4. In order to classify pattern x we look for matching rules in the set $FOR(C)$ (the left condition is fulfilled by its attributes).

5. If there is only one matching rule, then we classify this pattern x to the class which is indicated by its decision attribute j, because for sure such rule is belonging to the lower approximation of all rules indicating j, this rule is certain.

6. If there is more then 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 we take the index of boundary region for which the strength of corresponding rule is maximal. In such a case we take into account the rules which are possible.

7. In other cases: no rule was found or few rules have the same strength factor 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 we cannot find patterns in the learning set so as a consequence we generate dummy rule, useless in the classification process. The main drawback of algorithm presented in that is starts with an arbitrary 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 pattern x is rejected. The modification is as follows:

1. Algorithm starts with an arbitrary chosen step of granulation. Generally, it is a high value to ensure that recursion can be invoked by decreasing G value. Shortly speaking, we divide every domain of feature into G intervals. The whole procedure 1-6 is repeated from the previous point.

2. If for the pattern x we cannot find neither certain nor possible decision rule, it means that algorithm cannot find proper representation in learning set. Then we try to find matching rule by increasing recursively the current interval for every condition attribute $l = 1, \ldots, q$ and the learning procedure is invoked once again until the proper rule is found.

3. Recursive algorithm stops if for every attribute $G = 1$. Then the decision is taken randomly.

4. To enhance the process of finding proper decision formula for different G the decision set $FOR(C)$ for the certain G are stored in the memory.

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 rule which will ensure proper classification. When we have no expert knowledge about dataset it is not an easy task to generate them on its own. In the literature one can find many practical examples of how to generate $IF-THEN$ rules, from the statistical tools to heuristic algorithms. In this thesis genetic algorithm will be used to generate rule set as one of the approach in the recent year (more information about genetic algorithm can be found in section 2.5). For the algorithm construction we have to make few assumptions:

- The same as for rough sets algorithm, we assume that we have N training patterns.

- A set F of linguistic values and their membership functions is given for describing each attribute.

I think that the second point requires in-depth explanation. First of all how to partition each attribute into linguistic values and how to describe each membership function. Here is the place for genetic algorithm to find optimal parameters.

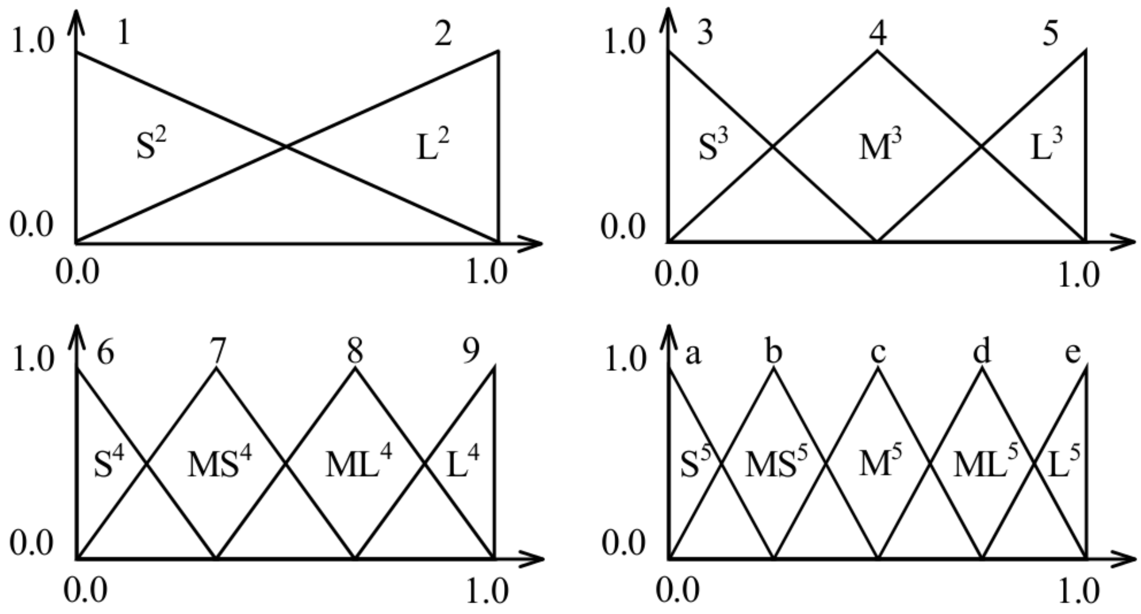


Figure 3.1: Example fuzzy partition set for an attribute

Fig. 3.1 shows an example how to generate fuzzy set F of possible membership functions. In this example 14 membership functions can be used. Each function has a subscript defining its linguistic value. To emphasize with which problem we facing with let take into account that each attribute is divided in the same way as depicted in Fig. 3.1. Having d-dimensional feature space possible combination of membership functions for rule generation is equal to 14^d. To evaluate each solution is computationally impossible in a reasonable time.

In this section we reconsider fuzzy rules in the following form:

$$IF\, x_1 = A_{q1}\, AND\, x_2 = A_{q2}\, AND\, \ldots\, AND\, x_d = A_{qd}\, THEN\, class\, C_q\, with\, CF_q$$

where $x = (x_1, x_2, \ldots, x_d)$ is a d-dimensional pattern vector; A_{qi} is an antecedent fuzzy set with a linguistic label (taking into account the example from 3.1 A_{qi} can be one label from the set $\{1, 2, \ldots, 9, a, b, \ldots, e\}$); C_q is a consequent class and can have value one of $\{1, \ldots, m\}$, CF_q is a rule strength determined in the training phase (see more in section 3.3.2); q determines the number of rule from the set

$\{1, \ldots, N_{rule}\}$, generally N_{rule} is about ten or twenty. Rule strength CF_q is used in the defuzzyfication process.

3.3.2 Rule generation

The process of rule generation for fuzzy logic without the expert knowledge is complex and done in couple steps. To construct fuzzy rule we use available training set. Let say that at first we generate randomly N_rule rules. For each training pattern x_p calculate the compatibility grade of a single rule connected with antecedent part $A_q = (A_{q1}, A_{q2}, \ldots, A_{qd})$ using the product operator of each membership function $\mu_{A_{qi}}$ determined for A_{qi}:

$$\mu_{A_q}(x_p) = \mu_{A_q 1}(x_p) \cdot \mu_{A_q 2}(x_p) \cdot \ldots \mu_{A_q d}(x_p) \tag{3.1}$$

If we know how to calculate the compatibility grade of each training pattern now we can determine C_q and CF_q for each rule. The fuzzy probability $P(class\, h|A_q)$ of class h, $h = (1, \ldots, m)$ is given by eq. (3.2)

$$Pr(class\, h|A_q) = \frac{\sum\limits_{x_p \in class\, h} \mu_{A_q}(x_p)}{\sum\limits_{p=1}^{m} \mu_{A_q}(x_p)} \tag{3.2}$$

For the rule R_q the label of class is assigned according to eq. (3.3)

$$R_q : C_q = max_{h=\{1,\ldots,m\}}\{Pr(class\, h|A_q)\} \tag{3.3}$$

In the learning phase it can happen that that rule R_q can be activated by patterns coming from different classes. We have to determine the strength factor for this rule if we have chosen a proper class h for rule R_q.

$$R_q : CF_q = Pr(class\, h|A_q) - \sum_{h=1, h\neq C_q}^{M} Pr(class\, h|A_q) \tag{3.4}$$

If CF_q in eq. (3.4) is negative then rule R_q is denoted as dummy and not used in later reasoning.

3.3.3 Fuzzy reasoning

Let assume that N_{rule} fuzzy rules are generated with indicators C_q, CF_q determined by eq. (3.3, 3.4). Then the process of classification is done as follows:

$$\Psi(S, x_p) = C_q \leftarrow max_{h=\{1,\ldots,M\}}\{\mu_{A_q}(x_p) \cdot CF_q\} \tag{3.5}$$

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_q.

If multiple fuzzy rules have the same maximum product μ_{A_q} 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
- Triangular membership functions are used and described by 2-tuple (a, b), where a is the center value, and b determines left and right extend of function respectively.

$$f(x) = \begin{cases} \frac{-1}{b} \cdot x + \frac{a+b}{b} & x \geq a \, and \, x \leq (a+b) \\ \frac{1}{b} \cdot x - \frac{a-b}{b} & x \geq (a-b) \, and \, x < a \\ 0 & otherwise \end{cases} \tag{3.6}$$

- 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 Mitchigan approach was 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_q. 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 and that for each attribute 14 membership functions plus one variable telling to omit the attribute (called

$DON'T\,USE$). 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

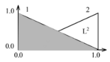
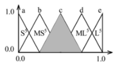
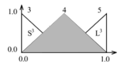
$IF\ x_1\ IS$  $AND\ x_2\ IS$  $AND\ x_4\ IS$  $THEN\ C_q = 1\ CF_q = 0.85$

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. Firstly the nature of pattern recognition task must be taken into account and secondly the structure of the fuzzy algorithm. Generally, we aiming at generating rule with the highest CF_q grade, smallest number of attributes and the highest classification rate. Fitness function is given by eq. (3.7)

$$F_{fg} = w_1 \cdot NC + w_2 \cdot NNC + (\frac{1}{NOF})^2 + w_3 \cdot CF \tag{3.7}$$

    where w_1, w_2 are weights for a reward and punishment to the rule based on the classification result; NC, 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. The best individuals are those which maximize function F_{fg}

- Cross-over, mutation and population generation
  - From the whole population two individuals were chosen randomly to constitute father and mother parent. With a probability of 0.5 each allele was picked either from mother or father chromosome. In this way we generate two new individuals.
  - In the particular generation one chromosome is chosen randomly and later in each allele new membership function is taken from other possible function. For example is 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. 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 is greater than the average $F \geq F_{avg}$

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

In this section the main algorithm in this thesis will be described.

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 hybridization of rough sets and fuzzy logic is presented. The next subsections show algorithm construction.

3.4.2 Rough sets and genetic algorithm

Rough sets algorithm presented in section 3.2 uses an arbitrary chosen step of granulation which affects the accuracy of classification. Additionally, each attribute uses the same granulation interval which in some cases gives good results in other efficiency is low. Another drawback of the basic rough set algorithm is that it uses all attributes for rule construction.

Finding the optimal attribute reduct and rough set partition is NP problem. To overcome this obstacle a genetic algorithm was 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 for the 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 not used in 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. (3.8) is used.

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

where w_1, w_2 are weights for a reward and punishment to the individual on the classification result; NC, 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 rule which are derived from partition given by the individual; w_3 is the weight.

The whole procedure can be summarized in few steps:

1. Determine 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
3. Treat each individual as a rough set partition and calculate lower, upper approximation and 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

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

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

1. Rough sets classifier construction using genetic algorithm presented in section 3.2
2. Fuzzy logic classifier construction with rule generation by heuristic approach described in 3.3.4
3. Pattern recognition using hybrid classifier

Each step plays an important role in the whole process and affect the final classification accuracy. 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 rule were found or more than one rule have the same strength, but different label then pattern is rejected and processed by fuzzy logic classifier.

Illustrative scheme of hybrid classifier is presented in fig. 3.3

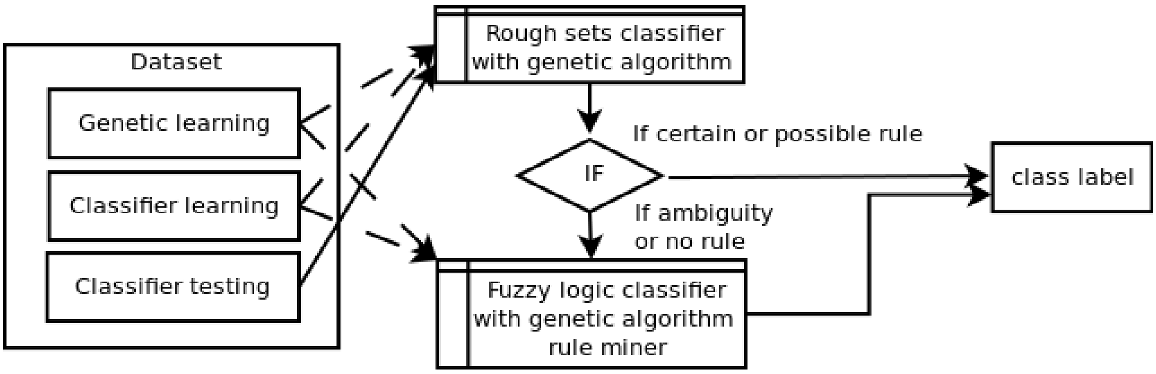


Figure 3.3: Schematic diagram how hybrid classifier works

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 algorithm were written:

1. Basic Rough sets algorithm
2. Rough sets algorithm with modification of granulation step
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
    - Fuzzy logic algorithm (the same as in point 3)

4.2 Datasets

To perform all the experiment datasets from *UCI* repository were used. 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 goal in choosing an appropriate datasets was to ensure diversity and test algorithms with complicated and complex problems. Below each dataset is described in more details:

- 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: 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 the 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 will be listed methods of algorithm fitness scoring

- An absolute error which is the distance between the solution found by algorithm and known a priori global minimum. There are distinguished three variants of this indicator
  - The best value from n probes

$$B = MIN\,|f^*(\underline{x}) - f(\underline{x})| \tag{4.1}$$

  - The worst value from n probes

$$W = MAX\,|f^*(\underline{x}) - f(\underline{x})| \tag{4.2}$$

  - The average value from n probes

$$A = \frac{1}{n}\sum_{n=1}^{n} |f^*(\underline{x}) - f(\underline{x})| \tag{4.3}$$

  where $f^*(\underline{x})$ is the value found by algorithm and $f(\underline{x})$ is a priori global minimum.

- Error variance from n simulations

$$\sigma^2 = \frac{1}{n-1}\sum_{i=1}^{n} [A - B]^2 \tag{4.4}$$

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 parameter, they were written to the file so that easily change their value in testing procedure. Output results(efficiency indicators) were written to CSV for further processing. The most preferable environment for running this project is Eclipse, free to download from the Internet.

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 problems 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). 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 holdout 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. To ensure that presented results are reliable each test was repeated 10 times.

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. At first the number of intervals for each attribute is the same and denoted as K_l, where $l \in (1, \ldots, d)$. 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 have the same strength but different class label.
- C^{*}, P^{*}, V^{*} - total number of certain, possible, void 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 γ was equal to one.

Analyzing the results of simulation presented in table 5.1 one can see that the quality of the algorithm depends on the step of granulation G and better results are obtained rather for small G. It can be concluded that decreasing the step of granulation causes that the number of decision formulas with the strength equal to zero is growing. The level of discretization is correlated with the number of cells in which decision was certain or possible because of predominance of one class. In a situation with a small number of fields there were areas with the same number of object from both class, but small number of void cells. Decreasing the step of granulation results in more fields where decision was certain for one class, but at the same time it was noticeable that the number of areas without representative is increasing.

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	114	0/0	125/95	28/19	3	7	54
5	114	0/0	121/97	32/17	2	4	119
6	114	3/1	65/51	85/62	5	8	203
7	115	0/0	91/74	62/41	3	6	334
8	113	2/1	24/17	127/95	8	9	495
9	114	4/4	59/48	90/62	6	7	716
10	112	36/25	38/32	79/55	12	13	975
11	112	5/2	30/23	118/87	7	9	1315
12	111	49/41	9/4	95/66	19	12	1697
13	109	0/0	13/9	140/100	17	12	2168
14	112	9/6	28/24	116/82	24	14	2706
15	105	30/25	17/4	106/76	27	13	3335
16	108	2/1	20/17	131/90	22	12	4062
17	100	3/1	38/24	112/75	24	17	4872
18	101	2/1	11/8	140/92	25	13	5794

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

In the previous section (5.2.2) it was shown that the granulation step strongly affect the classification accuracy. Greater G implies that we have more certain or possible rule, but on the other 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 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 G is decreased by $\epsilon = 1$ until proper certain for possible solution is found. The same dataset was used as in the previous one with the same algorithm settings. Results of classification are presented in fig. 5.1 for the basic rough sets (blue line-BRS) and algorithm with modification of decision rules (orange line-MRS).

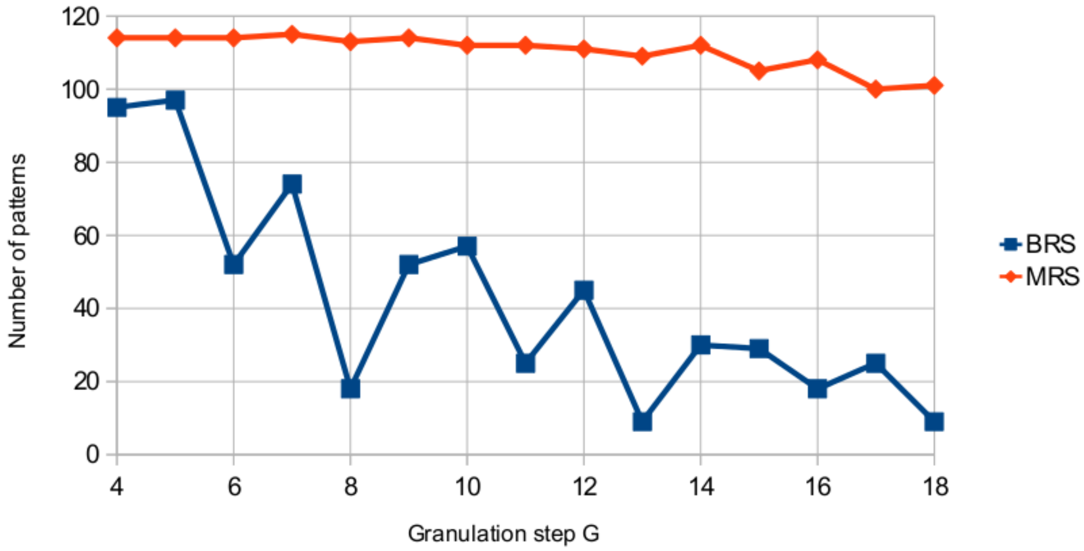


Figure 5.1: Comparison of Basic Rough sets algorithm with algorithm where modification of decision rules is introduced.

Looking at fig. 5.1 one can observer that modification of granulation step G increases the classification accuracy and even if the granulation step is changed 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. Additionally, 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 algorithms 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 decision rules are long and tangled. Additionally, some features are useless in classification, instead of valuable information bring noise to the system and deteriorate final results.

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

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. As the input we have dataset without no expert knowledge of how to appropriately divide feature space into fuzzy sets, how many membership functions we need. The goal is to 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 dataset which is iris dataset with four attributes. At the beginning each feature was divided into 14 membership functions in the same way as presented in fig. 3.1 The final rule set which was able to classify 32 out of 34 testing pattern is presented in fig. 5.2:

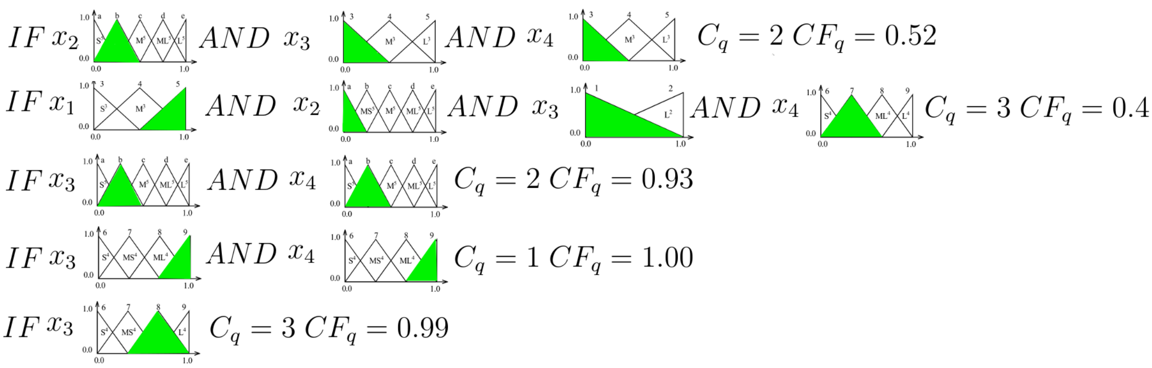


Figure 5.2: Example of rule set generated for Iris dataset

Analyzing figure 5.2 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.

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)

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

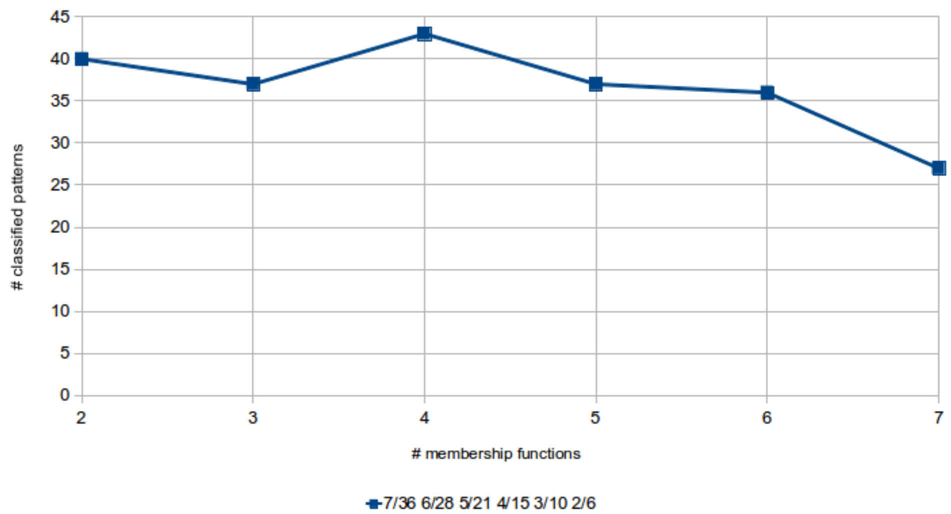


Figure 5.3: Impact of initial number of membership functions MF per attribute on the final fuzzy classification rate

Looking at the fig. 5.3 one can conclude that it is not worth increasing the number of membership functions (MF) per attribute because for greater MF algorithm obtained worse results in case of classification accuracy. From conducted simulations it was concluded that the optimal k value is 4. If not implicitly stated, $k = 4$ will be used in next tests.

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), where the notation is as follows:

- A number of attributes describing pattern in dataset
- O number of patterns for classification
- 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
- AU number of features used by genetic rough sets algorithm for classification.

Results of simulation are presented in table 5.2. 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 5 and for algorithm with modification of decision rules starting granulation value was 7.

Dataset	A	O	RSR	$RSMR$	GRR	AU
haberman	3	76	49	58	61	2
iris	4	39	32	38	39	2
bupa	6	86	28	53	57	3
pima	8	192	69	126	155	5
wine	13	45	0	15	44	2

Table 5.2: Accuracy of classification for genetic rough sets and basic rough sets algorithms for different datasets

From table 5.2 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 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 solution can compete with other classifiers. As the source of reference different types of classifiers were chosen to ensure the greatest diversity:

- LDAC classifier (Linear 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. 5.4)

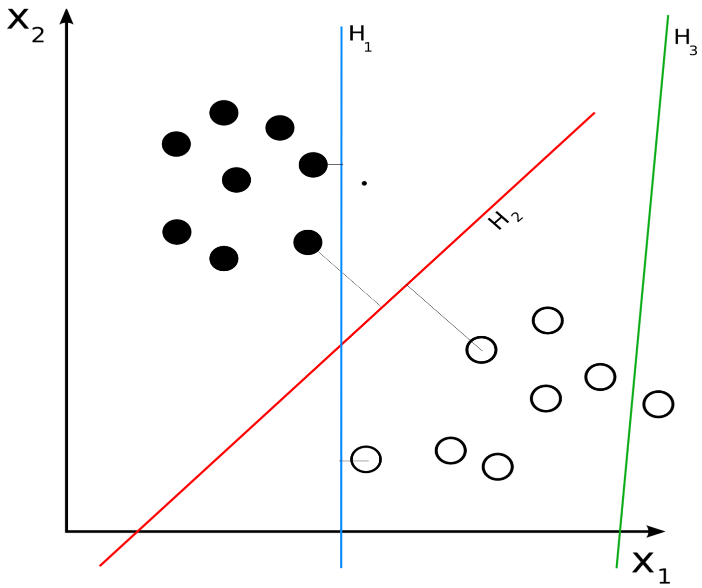


Figure 5.4: Example of linear discriminant classifier for 2-dimensional problem

- 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 path indicated 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 5.3, where O determines the number of patterns for classification. Last row in table shows results for hybrid classifier described in section 3.5. Numbers in bold font indicates which classifier obtained the best result for a particular dataset.

Table 5.3: Comparison of hybrid rough fuzzy classifier with other common classifiers

Algorithm	iris	bupa	pima	haberman	wine	wdbc	thyroid
O	39	86	192	76	45	142	55
LDAC	38	62	146	58	44	**138**	51
3-KNN	38	56	131	56	32	133	52
Gini Index	38	56	123	58	41	131	53
Max likelyhood	38	51	145	58	45	135	55
Svm	26	**63**	141	56	43	133	53
Hybrid	**39**	59	**155**	**61**	**45**	135	**55**

From table 5.3 one can conclude that hybrid classifier obtains quite good results comparing with other classifiers. From seven datasets five 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 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 was still 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 for each attribute. Conducted test confirmed that proposed algorithm can compete with other classifiers and obtains reliable results. Next paragraph will shortly will shortly summarize each research.

6.2 General conclusions

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 thesis to evaluate classifiers well-known datasets from *UCI* Repository. 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 or 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, 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 [**?**]

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

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

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

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

[5] 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

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

[7] 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

[8] 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

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

[10] 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

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

[12] 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

[13] 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

List of Figures

List of Tables

Appendix A Program description