Fuzzy Miner
A Fuzzy System for Solving Pattern Classification Problems

Nikos Pelekis[1,2], Babis Theodoulidis[1], Ioannis Kopanakis[1]

[1]Center of Research in Information Management
Department of Computation, UMIST[†]
URL: http://www.crim.org.uk
E-mail: pele@ath.forthnet.gr, babis@co.umist.ac.uk, kopanak@csd.uoc.gr

[2]Department of Informatics
University of Piraeus
Athens, Hellas
URL: http://www.unipi.gr
E-mail: npelekis@unipi.gr

Abstract. The purpose of this paper is to study the problem of pattern classification as this is presented in the context of data mining. Among the various approaches we focus on the use of Fuzzy Logic for pattern classification, due to its close relation to human thinking. More specifically, this paper presents a heuristic fuzzy method for the classification of numerical data, followed by the design and the implementation of its corresponding tool (*Fuzzy Miner*). The initial idea comes from the fact that fuzzy systems are universal approximators of any real continuous function. An approximation method coming from the domain of fuzzy control is appropriately adjusted into pattern classification and an "adaptive" procedure is proposed and developed for deriving highly accurate linguistic if-then rules. Extensive simulation tests are performed to demonstrate the performance and advantages of Fuzzy Miner, as well as its potential commercial benefits over a real world scenario.

1. Introduction

Recently, our capabilities of both generating and collecting data have increased rapidly. Consequently, data mining has become a research area with increasing importance. Data mining also referred to as knowledge discovery in databases [2], deals with problems such as characterization, comparison, association, classification, prediction and clustering. This paper elaborates with the problem of classification. Broadly speaking, pattern classification (or recognition) is the science that concerns the description or classification of measurements. More technically, pattern classification is the process that finds the common properties among a set of objects in a database and classifies them into different classes, according to a classification model.

Classical models usually try to avoid *vague, imprecise* or *uncertain* information, because it is considered as having a negative influence in an inference process. This paper accepts the challenge to deal with such kind of information, by introducing a fuzzy system, which deliberately makes use of it. The main idea of fuzzy systems is to extend the classical two-valued modelling of concepts and attributes like *tall, fast* or *old* in a sense of gradual truth. This means that a person is not just viewed as *tall* or *not tall,* but as tall to a certain degree between 0 and 1. This usually leads to simpler, more suitable models, which are easier to handle and are more familiar to human thinking. This paper, after providing a brief

[†] PO Box 88, Sackville Street, Manchester, M60 1QD, UK
Tel: +44 161 200 3309, Fax: +44 161 200 3324

comparative overview of pattern classification approaches (section 2), follows the above paradigm and proposes an effective heuristic fuzzy method for the classification of numerical data (section 3). The initial idea comes from the fact that fuzzy systems are universal approximators [4] of any real continuous function. Such an approximation method [8] coming from the domain of fuzzy control systems is appropriately adjusted in order to produce a powerful working solution in the domain of pattern classification. An "*adaptive*" process is also introduced, developed and incorporated into the previous mechanism for deriving automatically highly accurate linguistic if-then rules. The description of the methodology is combined with the illustration of the design and the implementation issues of the corresponding tool (*Fuzzy Miner*). The current work is evaluated (section 4) by extensive simulation tests. Finally, the paper concludes (section 5) and identifies promising directions for future work pointed to by this effort.

2. Comparative Overview of Pattern Classification systems

Already, when the field was still in its very infancy, it was realized that statistics and probability theory had much to offer to pattern classification [11]. The question of whether or not a given pattern "belongs" to some pattern class may naturally be treated as a special case of the statistical decision theory problem. Effective though as it is, the statistical approach has built-in limitations. For instance, the theory of testing statistical hypotheses entails that a clear-cut yes or no answer should always decide upon the membership of a pattern in a given class. Clearly, not all of the real life patterns admit of such coarse decisions. Sometimes information in a pattern is not simply in the presence or the absence of a set of features, but rather the interconnection of features contains important structural information. Indeed this relational information is difficult or impossible to be quantified by a feature vector form. This is the underlying basis of structural pattern classification. Structural based systems assume that pattern structure is quantifiable. As such, complex patterns can be decomposed recursively in simpler subpatterns in almost the same way that a sentence can be decomposed in words. The analogy directed researchers toward the theory of formal languages. The process that results in an answer to a classification question is called *syntax analysis* or *parsing.*

Fuzzy logic is a superset of conventional (Boolean) logic that has been extended to handle the concept of partial truth (values between "completely true" and "completely false") [17]. Fuzzy Pattern Classification is one way to describe systems and the behaviour of systems. A system can be described by using adjectives like "high", "mid", "low". Pattern Classification using fuzzy logic [6, 15], partitions the input space into categories (pattern classes) $w_1, \ldots, w_n$ and assigns a given pattern $v = (v_1, v_2, \ldots, v_n)$ to one of those categories. If v does not fit directly within a category, a "goodness of fit" is reported. By employing fuzzy sets as pattern classes, it is possible to describe the degree to which a pattern belongs to one class or another. By viewing each category as a fuzzy set and identifying a set of fuzzy IF-THEN rules as assignment operators, a direct relationship between the fuzzy set and pattern classification is realized. The main advantage of the approach is the close relation to the human thinking. On the other hand, the disadvantages are the fact that a fuzzy system cannot learn from data, and that there is no formal method to tune the membership functions.

Fuzzy, statistical and structural approaches are valid approaches to the classification problem. The point is that probability (statistical approach) involves crisp set theory and does not allow for an element to be a partial member in a class. Probability is an indicator of the frequency or likelihood that an element is in a class. On the other hand, formal grammars (structural approach) have a difficulty in learning structural rules. Finally fuzzy set theory deals with the similarity of an element to a class. As such, if we were to classify someone as "senior", fuzzy membership makes much more sense than probability. On the contrary, if we were to classify the outcome of a coin flip, probability is preferable.

The course of argumentation followed so far puts the pattern classification theme into a technical-mathematical framework. Since pattern classification is an ability of intelligent natural systems, it is possible to imitate the neuron - the basic unit of the brain - by an analogue logical processing unit, which processes the inputs and produces an output, which is either on or off. Thus by extension, a simple neuron can classify, the input in two different classes by setting the output to “1’, or “0”. The neuron is very good to solve linearly separable problems, but fails completely to solve apparently simple problem such as the XOR one. This issue is easily overcome by multilayer neurons that use more than one neuron and combine their outputs into other neurons, which would produce a final indication of the class to which the input belongs [1].

Among the above-mentioned solutions, fuzzy logic and neural networks can be an answer to the vast majority of classification problems. Both approaches attempt to determine the transfer function between a feature space and a given class and can be automatically adapted by the computer in an attempt to optimize their classification performance. One difference between the two methods is that the membership functions of a fuzzy classifier can be initialized in a state close to the correct solution. What this means is that a fuzzy classifier can be set up by a skilled designer to do a pretty good job of classification even before the classifier is adjusted by the computer. A neural network, however, can only learn from scratch, and as such, can only be initialized in a random state. But their learning capabilities are significant as different learning algorithms are available and they have great potential for parallelism, since the computations of the components are largely independent of each other. But drawbacks are, the impossibility to extract rules from neurons for interpretation, and that prior knowledge cannot be used to initialize the system. As such, the training of the computer to optimize the classifier is usually much faster with a fuzzy classifier than a neural network. Consequently, combining fuzzy logic and neural networks (neuro-fuzzy systems) we can avoid the drawbacks of each method. While the learning capability is an advantage from the viewpoint of a fuzzy system, from neural network side, there are additional advantages to a combined system. We can initialize the system by establishing rules and membership functions and thus shorten the learning process. The result is obtained by modification of the rule base or the membership functions, allowing its interpretation as a fuzzy system.

Finally, in many applications of fuzzy rule-based systems, fuzzy if-then rules have been obtained from human experts. Recently, various methods were proposed for automatically generating fuzzy if-then rules from numerical data. Most of these methods have involved iterative learning procedures or complicated rule generation mechanisms such as gradient descent learning methods [7], genetic-algorithm-based methods [5], [9], least-squares methods [12], a fuzzy c-means method [13] and a neuro-fuzzy method [14]. In [16], an efficient rule generation method with no time-consuming iterative procedure is proposed and its high performance is demonstrated.

3. Description of Fuzzy Miner

Fuzzy rule-based systems have as theoretical base the theory of *Fuzzy Logic.* Fuzzy set theory [17] provides a strict mathematical framework in which vague conceptual phenomena can be precisely and rigorously studied. In this section, we describe a simple but powerful fuzzy system for solving pattern classification problems and we provide the reader with a brief description of the components of the Fuzzy Miner, their internal processes and their interrelationships. The reader interested in a detailed description of the design and implementation issues of Fuzzy Miner is referred to [10]. The preliminary work has mainly been focused on the study and understanding of a method proposed in [8], which is heuristic method for automatically generating fuzzy if-then rules from numerical data. Fuzzy if-then rules with nonfuzzy singletons (i.e., real numbers) in the consequent parts are generated by the proposed heuristic method. The main advantage of these fuzzy if-then rules is the simplicity of a fuzzy reasoning procedure because no defuzzification step is required. In the proposed heuristic method, the consequent real number of each fuzzy if-then rule is determined as the weighted mean value of given numerical data. Thus, the proposed heuristic method does require neither time-consuming iterative learning procedures nor complicated rule generation mechanisms.

3.1. Design & Architecture of the fuzzy rule-based system

Fuzzy rule-based systems are also known as fuzzy inference systems, fuzzy models, fuzzy associative memories (FAM) or fuzzy controllers. Basically, such fuzzy rule-based systems are composed of four principal components: a fuzzification interface, a knowledge base, a decision-making logic and a defuzzification interface. Fuzzy Miner employs this architecture depicted in figure 1.

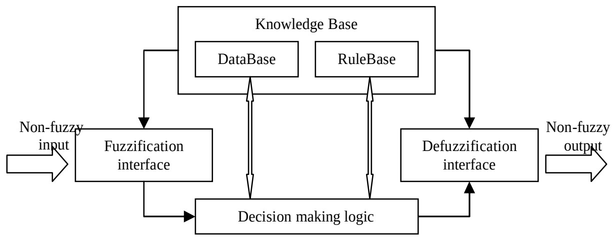


Figure 1 Architecture of Fuzzy Miner

The initial algorithm [8] considers a single-output fuzzy rule-based system in the *n*-dimensional input space $[0, 1]^n$, so just for simplicity reasons we keep for the moment these assumptions. The actual algorithm implemented introduces a multiple-output fuzzy rule-based system with optional task, the mapping of the input spaces to the $[0, 1]^n$ space (normalization process). Of course, when normalization process is selected an appropriate action is performed after the end of the algorithm to map reversely the normalized data to their primitive spaces. Let us assume that the following *m* input-output pairs are given as training data for constructing a fuzzy rule-based system:

$$\{(x_p;y_p) \mid p = 1, 2, \ldots, m\}, \tag{3.1}$$

where $x_p = (x_{p1}, x_{p2},\ldots, x_{pm})$ is the input vector of the *p*th input-output pair and y_p is the corresponding output.

3.1.1. Fuzzification interface

The fuzzification interface performs a mapping that converts crisp values of input variables into fuzzy singletons. Basically, a fuzzy singleton is a precise value and hence no fuzziness is introduced by fuzzification in this case. This strategy, however, has been widely used in fuzzy system applications because it is easily implemented. Here we employ fuzzy singletons in the fuzzification interface.

3.1.2. Knowledge base

The knowledge base of a fuzzy rule-based system consists of two components, i.e., a *database* and a *rule base.*

***Database* -** There are two factors that determine a database, i.e., a fuzzy partition of the input space and membership functions of antecedent fuzzy sets. Fuzzy Miner in order to develop the appropriate infrastructure defines three corresponding objects, namely *Database*, *Fuzzy Partition* and *Membership Function*. *Database* object provides a complete set of functionalities upon the data (e.g. normalization/denormalization process) that the algorithm needs in order to operate effectively. Someone can think of a Database object as the realization of a real database, which enables us to store, retrieve, update and generally manipulate data. Database object is defined as a 2D array, where the first dimension corresponds to the row of a database table and the second dimension corresponds to the column (input-output space).

We assume that the domain interval of the *i*th input variable x_i is evenly divided into K_i fuzzy sets labelled as $A_{i1}, A_{i2}, \ldots, A_{iK_i}$ for $i = 1, 2,\ldots,n$. Then the *n*-dimensional input space is divided into $K_1K_2 \ldots K_n$ fuzzy subspaces:

$$\left(A_{1j_1}, A_{2j_2}, \ldots, A_{nj_n}\right),\ j_1=1, 2,\ldots, K_1; \ldots;\ j_n=1, 2,\ldots, K_n. \tag{3.2}$$

For example, in the case of a two-dimensional input space, the fuzzy subspace $\left(A_{1j_1}, A_{2j_2}\right)$ corresponds to the region shown in figure 5(a). Figure 5(b) shows an example of the fuzzy partition for K_1 = 5 and K_2 = 5 in the case of a two-input single-output fuzzy rule-based system.

Membership Function object can be perceived as the mean to measure the degree of compatibility of a data value to a fuzzy set, or as the probability that this data value "belongs" to a fuzzy set. Because we wanted to be able to use more than one membership functions, we adopted a generic representation that enables the definition of different kinds of membership functions. As such, the user of the fuzzy classifier can use not only triangular membership functions, but also trapezoidal and bell-shaped. In order to represent a triangular fuzzy membership function, three parameters are enough. However, from a practical point of view, to use trapezoidal and/or bell-shaped (Gaussian) membership functions, four parameters are necessary. Below we can see all the types of membership functions that Fuzzy Miner supports.

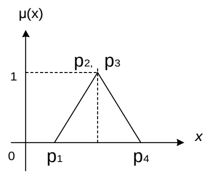


Figure 2 Triangular

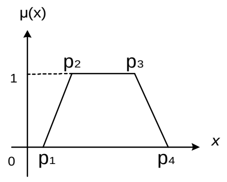


Figure 3 Trapezoidal

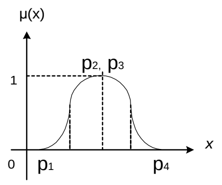
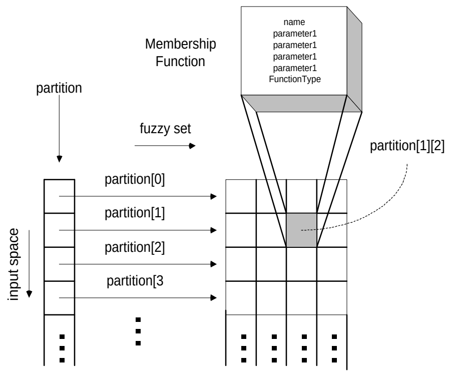
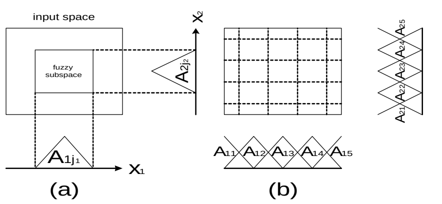


Figure 4 Bell-shaped

Fuzzy Partition object supports the notion that input and output spaces should be partitioned to a sequence of fuzzy sets. Each of these fuzzy sets has a description of its membership function. Normally there should be one Fuzzy Partition object per input and output space, but just for simplicity reasons we make the assumption that the object Fuzzy Partition represents all the fuzzy partitions. We further assume that all the fuzzy partitions are composed of the same number of fuzzy sets N. As such the object Fuzzy Partition is a 2-D array of Membership Functions (figure 6). The first dimension corresponds to the input space number and the second dimension corresponds to the fuzzy set number. Note that it is necessary to use a different fuzzy partition for each input space because the domain intervals of the input variables may be different.

The main functionality that Fuzzy Partition object offers to Fuzzy Miner is taking place by the time of its construction and it is the actual fuzzy partitioning. Analytically, in order to create the object Fuzzy Partition, the domain intervals of the input and output variables are needed. The domain interval of a variable x_i is taken as [x_{imin}, x_{imax}], where x_{imin} and x_{imax} are the minimum and maximum of the variable in the training data set. Furthermore, although the fuzzy partition of an input space is only supposed to cover the domain interval of the input variable, the case of input values lying outside the domain interval must be taken into account. As shown in figure 7, where we present the partitioning in the case of triangular membership function, by assigning the value $-\infty$ to the two first parameters of the first fuzzy set and the value $+\infty$ to the two last parameters of the last fuzzy set, the fuzzy partition corresponding to an input variable x covers $\Re$.





input space
fuzzy subspace
x_2
A_{2j_2}
A_{1j_1}
x_1
(a)
(b)
A_{11} A_{12} A_{13} A_{14} A_{15}
A_{21} A_{22} A_{23} A_{24} A_{25}

Figure 5 (a) Fuzzy subspace and (b) Fuzzy partition for K1 = 5 and K2 = 5

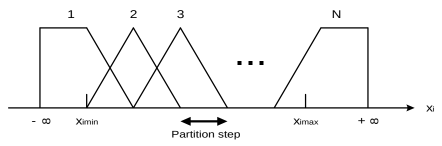


Figure 6 Fuzzy partition structure

Figure 7 Fuzzy partitioning for triangular MF

Rule base **-** The rule base consists of a set of fuzzy if-then rules in the form of 'IF a set of conditions is satisfied, THEN a set of consequences can be inferred". We assume that the rule base is composed of fuzzy if-then rules of the following form:

Rule $R_{j_i \dots j_n}$: If x_1 is A_{1j_1} and … and x_n is A_{nj_n} then y is $b_{j_1 \dots j_n}$, j$_1$=1, 2,…, K$_1$; …; j$_n$=1, 2,…, K$_n$, (3.3)

where $R_{j_i \dots j_n}$ is the label of each fuzzy if-then rule and $b_{j_1 \dots j_n}$ is the consequent real number. These fuzzy if-then rules are referred to as simplified fuzzy if-then rules and have been used in [3], [7] and [9]. For determining the consequent real number $b_{j_1 \dots j_n}$ of the fuzzy if-then rule $R_{j_i \dots j_n}$ in (3.3), let us define the weight of the *p*th input-output pair *(x_p;y_p)* as

$$W_{j_1 \dots j_n}(x_p) = \{m_{j_1 \dots j_n}(x_p)\}^a, \tag{3.4}$$

where *a* is a positive constant. The role of the positive constant *a* will be demonstrated by computer simulations. Using the weight $W_{j_1 \dots j_n}(x_p)$ of each input-output pair, we propose the following heuristic method (the weighted mean value of y_p's) for determining the consequent real number:

$$b_{j_1 \dots j_n} = \sum_{p=1}^{m} W_{j_1 \dots j_n}(x_p) \cdot y_p \Big/ \sum_{p=1}^{m} W_{j_1 \dots j_n}(x_p) \tag{3.5}$$

Rulebase is the main component of the application and supports all the functionality that we need, in order to implement the various aspects of Fuzzy Miner. It generates the fuzzy rules from training data and furthermore is responsible for the decision making part of the algorithm (see section 3.1.3). An additional task that is supported by our rule generation method is that of an *adaptive procedure*, which expands a given rulebase, during the processing of testing data when the inference engine (decision making) of the algorithm is running. A *Rulebase* object is implemented mainly as an array of *Rules* that in its turn is represented as an array of integers, corresponding to the conditional part and an array of *Then Part* objects, corresponding to the consequent part, one element per output space. *Then Part* objects are needed in order to calculate the consequent parts of a fuzzy rule (the relatively complex fraction (nominator / denominator) of equation 3.5). The computational development of the above mathematically described process for inferring fuzzy rules, after given learning data and information concerning the number of inputs and outputs of these data is presented in figure 8:

***Adaptive procedure* -** Before illustrating how the decision-making method has been implemented, we introduce a simple procedure with which we expand the initial approach, for updating a rule base, which is called "adaptive" procedure. This procedure takes place concurrently with the decision making process, namely when testing data are examined, inferred output are calculated and are mapped to classes. The approach is based on an advantage of the fuzzy-numerical methods, which is the facility to modify a fuzzy rulebase, as new data become available. More specifically, when a new data pair becomes available, one rule is created for this data pair and is either added to the rule base, or, if a similar rule (same conditional part) already existed in the rule base, the existing rule is updated. By this we mean that the consequent part of the existed rule is improved, by applying the generation method one more time for this specific conditional part. Thus, by using this "adaptive" procedure, which gives to Fuzzy Miner incremental characteristics, all available information is used, so decision making on testing data has better results.

```
Generate rules(numberOfInputs, numberOfOutputs, startOfLearnData, endOfLearnData)
{
        currentRule = 0;
        allocate memory for rulebase[currentRule];
        for all learning data pairs f
                usedData[f] = false;
        create temporary rule;

        for (i = startOfLearnData; i <= endOfLearnData; i++)
        {
                if (!usedData[i])
                {
                        construct rulebase[currentRule];
                        set IF part of rulebase[currentRule];
                        set THEN part of rulebase[currentRule];
                        calculate weight of rulebase[currentRule];
                        for all outputs j
                                numerator[j] = weight * (THEN part of rulebase[currentRule]);
                                denominator[j] = weight;
                        for ( j = i + 1; j <= endOfLearnData; j++)
                        {
                                set IF part of temporary rule;
                                set THEN part of temporary rule;
                                calculate weight of temporary rule;
                                if (!usedData[j] & currentRule has same IF part as temprule)
                                {
                                        for all outputs
                                                update numerator[k];
                                                update denominator[k];
                                        usedData[j] = true;
                                }
                        }
                        for all outputs
                                set THEN part of ruleBase[currentRule];
                        currentRule++;
                }
        }
}
```

Figure 8 Rule Generation Method

***Linguistic representation* –**In real-world applications, it may be desired that linguistic rules are generated from numerical data. In [13] an approach in proposed for deriving linguistic rules from fuzzy if-then rules with fuzzy sets in the consequent parts. Here another similar approach is followed for translating fuzzy if-then rules with consequent real numbers into rules, whose "then" part is a linguistic label and corresponds to the classification of the respective data pairs. In this connection, this approach can derive classification rules from fuzzy if-then rules with consequent real numbers, which may be generated by other rule generation methods as well as the described heuristic method. Let us assume that fuzzy if-then rules in (3.3) are given. To translate consequent real numbers into linguistic labels, suppose that the domain interval of an output *y* is divided into *N* fuzzy sets (i.e., linguistic labels) $B_1, B_2, \ldots, B_N$, which are associated with the membership functions $m_{B_1}, \ldots, m_{B_N}$, respectively. For example, these fuzzy sets may have linguistic labels such as S: small; MS: medium small; M: medium; ML: medium large and L: large. In this method, the given fuzzy if-then rules in (3.3) are transformed to the following fuzzy if-then rules:

Rule $R^*_{j_i \ldots j_n}$: If x_1 is A_{1j_1} and … and x_n is A_{nj_n} then y is $B^*_{j_1 \ldots j_n}$, with $CF^*_{j_1 \ldots j_n}$, (3.6)
j1=1, 2,…, K1; …; jn=1, 2,…, Kn,

where $B^*_{j_1 \ldots j_n}$ is the consequent fuzzy set characterized by the following membership function:

$$m_{B^*_{j_1 \ldots j_n}}\left(b_{j_1 \ldots j_n}\right) = \max\left\{m_{B_i}\left(b_{j_1 \ldots j_n}\right) \mid i = 1,2,\ldots,N\right\} \tag{3.7}$$

and $CF^*_{j_1 \ldots j_n}$ is the degree of certainty defined as

$$CF^*_{j_1 \ldots j_n} = m_{B^*_{j_1 \ldots j_n}}\left(b_{j_1 \ldots j_n}\right) \tag{3.8}$$

3.1.3. Decision making logic

The decision-making logic is the kernel of a fuzzy rule-based system, which employs fuzzy if-then rules from the rule base to infer the output by a fuzzy reasoning method. In this paper, we employ the following fuzzy reasoning method to calculate the inferred output of the fuzzy rule-based system. Given an input vector $x_p = (x_{p1}, x_{p2}, \ldots, x_{pn})$ the inferred output $y(x_p)$ is defined by

$$y\left(x_p\right) = \sum_{j_1=1}^{K_1} \cdots \sum_{j_n=1}^{K_n} m_{j_1 \ldots j_n}\left(x_p\right) \cdot b_{j_1 \ldots j_n} \Bigg/ \sum_{j_1=1}^{K_1} \cdots \sum_{j_n=1}^{K_n} m_{j_1 \ldots j_n}\left(x_p\right) \tag{3.9}$$

where $m_{j_1 \ldots j_n}\left(x_p\right)$ is the degree of compatibility of the input vector $x_p = (x_{p1}, x_{p2}, \ldots, x_{pn})$ to the fuzzy if-then rule $R_{j_i \ldots j_n}$ in (3.6), which is given by

$$m_{j_1 \dots j_n}\left(x_p\right) = m_{1j_1}\left(x_{p1}\right) \times \dots \times m_{nj_n}\left(x_{p_n}\right). \tag{3.10}$$

From (3.9), we can see that the inferred output *y(x$_p$)* is the weighted average of the consequent real numbers $b_{j_1 \dots j_n}$'s of the $K_1K_2 \dots K_n$ fuzzy if-then rules.

This method, given a testing data set, calculates the outputs of the Fuzzy Miner and performs a mapping from the inferred consequent real number to the respective fuzzy set (classification result) that this real number belongs to. Subsequently, this method stores both the original outputs and classifications of the testing data pairs and the inferred outputs with the resulted classifications to an output Database. What is more, in order to have better results, it utilizes the adaptive procedure, which is an embedded process and not a autonomous one. Finally, in order to evaluate the algorithm for the given testing data, decision-making method estimates the mean square errors, between the desired output y_p and the inferred output *y(x$_p$)*. This **P**erformance **I**ndex (PI) (see equation 4.1) and the number of unpredicted results are returned as results of the whole process.

3.1.4. Defuzzification interface

Basically, the defuzzification interface performs a mapping from the fuzzy output of a fuzzy rule-based system to a crisp output. The fuzzy rule-based system employed in this paper, however, does not require a defuzzification interface.

4. Evaluation of the Fuzzy Miner

This section focuses on examining the reliability and the validity of Fuzzy Miner. The process of classification is deterministic, meaning that the same input data will always produce the same result. As such, in order to measure the performance of the methods used to implement Fuzzy Miner, several experiments took place on all the different parameters that can lead in useful conclusions. For the experiments, we used the data set from the **A**thens **S**tock **E**xchange (ASE) market. The ASE data set keeps a vast amount of information concerning the daily transactions of the stock market of Greece. As has been already mentioned, the algorithm works with numerical data and fuzzy systems are universal approximators of any real continuous function. In order to take advantage of this important feature of fuzzy systems, and for the purposes of the evaluation, we design a classification task based upon the prediction/inference of a function that estimates a real number, which represents the degree of fluctuation of a stock price during a day. For further details the interested reader is referred to [10]. The primitive data set is restricted to those tuples from the database that concern transactions of banks stocks. In the following experiments, 3.000 input-output data pairs are used to assess the forecasting ability of the system. The sampling factor used to split these patterns was fifty percent, so the first 1.500 tuples are used for learning and the last 1.500 tuples for testing. Note that, since the fuzzy rule-based system can employ the "adaptive" approach, test data may also be learning data, although they do not participate in the creation of the initial fuzzy rule base.

Fitting & generalization ability for training and testing data - In order to evaluate the algorithm the summation of square errors is calculated, between the desired output y_p and the inferred output $y(x_p)$ for each input-output pair $(x_p; y_p)$. This performance index (PI) for Fuzzy Miner is given by the following equation:

$$PI = \sum_{p=1}^{m} \{y(x_p) - y_p\}^2 / 2 \qquad (4.1)$$

The two most important parameters of the fuzzy rule-based system are the value of factor *alpha* and the size of the fuzzy partitions. In order to understand the influence of these parameters on the PI, the algorithm has been invoked with different values of *alpha* varying from 0.1 to 50 and a fuzzy partition size varying from 2 to 25. The results of the simulations are not fully presented here due to space limitations (the reader can find them in [10]), but one can draw some conclusions that could be useful for someone that wishes to utilize Fuzzy Miner and obtain from it highly accurate results. The most obvious conclusion that someone could infer from these simulations is that larger sizes of fuzzy partition lead to better fitting (smaller PI) to the given input-output data pairs. The PI for both the original method and the method using the adaptive approach has been plotted against the number of fuzzy sets per fuzzy partition. Figure 9 depicts that Fuzzy Miner performs much better when it uses the adaptive approach and this is reasonable, as adaptive procedure is made to improve the approximation of the desired output. However, when a is bigger than one, the PI sometimes is becoming worse, due to the phenomenon of overfitting. What is more, when the number of fuzzy sets is high, the PI decreases very slowly, whereas for a small number of fuzzy sets, the PI is much more sensitive to the variation of the fuzzy partition size. Finally, the PI tends asymptotically towards the same limit as the fuzzy partition size increases. A second observation is that for each specific fuzzy set the PI decreases or increases depending on the value of *alpha*. More specifically, when a is less than five the PI is improving, but when it exceeds that limit the PI starts decreasing. The best fitting is presented when a is 5. As such, the PI of Fuzzy Miner can be improved by choosing an appropriate value of a. Figure 10 shows the desired output and two inferred outputs by the system, for two different values of a. When a is 5, is self-evident that the approximation of the formula is much better than when a is 0.1.

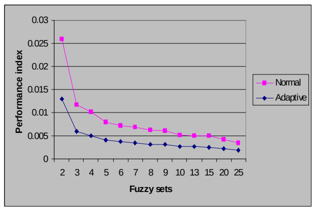


Figure 9 PI against size of fuzzy partitioning

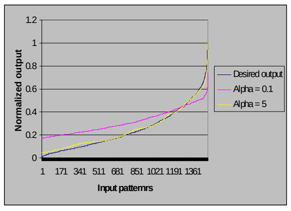


Figure 10 Fluctuation against alpha

***Classification success* -** From the same simulations, one can infer some useful conclusions for the usage and the classification power of Fuzzy Miner. First of all, when the number of fuzzy sets is fixed, then for values of *alpha* lower than five, the percentage of classification success increases as *alpha* approximates five. When it exceeds five the trend is either to stabilize or to decrease. This conclusion does not stand so strongly, as in the case of the PI. This is reasonable because, due to the vagueness that is introduced by the fuzzy sets, the PI of the classifier can be improved without a corresponding improvement of the classification success. Table 1 presents the trend of the classifier for the case of two fuzzy sets (classes).

Table 1 Classification accuracy against alpha

Alpha	0.1	0.5	1	5	10	50
Classification Accuracy	93%	94%	96%	97%	95%	93%

The previous reason is also the explanation why the percentage of success is the same for both the case of using the adaptive approach or not. Except for those few situations where the two percentages are identical, the general trend that is followed, is that for number of classes less than five, the adaptive approach gives higher classification results than the respective approach that is not using it. Unfavourably for fuzzy partition sizes more than five the phenomenon of overfitting does not allow to the adaptive procedure to provide always better performances. By overfitting in this situation, we mean that there are situations where the updating of the consequent parts of the fuzzy rules should not be performed if a predefined performance index is reached. Finally the strongest inference that someone can make is that the increment of the number of the classes, results in decrement of the percentage success of the classifier. This conclusion is described diagrammatically in figure 11, from where one can see that for low fuzzy partition sizes the classification success is reduced rapidly. On the contrary, for high number of fuzzy sets we observe stabilization in the rate of reduction of the classifier.

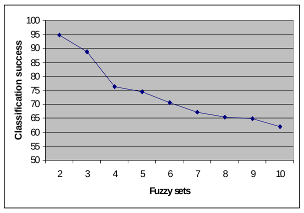


Figure 11 Classification success against size of fuzzy partitioning

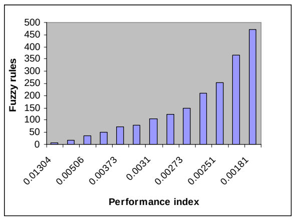


Figure 12 Size of rule base against PI

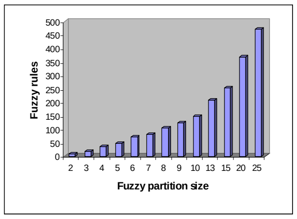


Figure 13 Rule base against fuzzy partition size

***Optimizing the size of the Rule Base* -** Another interesting observation is the variation of the performance index with respect to the rule base size. More specifically, figure 12 shows that when the PI decreases the number of the produced fuzzy rules is augmented in a stable rate. Using this graphical representation, it is possible to determine the "optimum" number of rules with respect to a performance requirement. Then, assuming a linear relation between the number of rules and the fuzzy partition size, the “optimum” number of fuzzy sets can also be determined. The relation between the fuzzy partition size and the inferred fuzzy rules is shown in figure 13. The fact that the number of the produced rules is more or less a linear combination of the number of fuzzy sets used to partition the input space, could also be inferred from table 2, where someone can see the number of rules for different sizes of fuzzy partition. Table 2 has an extra column containing the number of rules when the adaptive approach is applied. As expected in this situation the size of the rule base is bigger, as new rules are added during the decision making stage, where the testing data are processed.

Table 2 Produced rules for different numbers of fuzzy sets

Fuzzy set	2	3	4	5	6	7	8	9	10	13	15	20	25
Rules	9	17	37	49	71	81	106	124	147	209	254	367	470
Adaptive rules	9	20	41	61	81	94	136	161	190	293	355	532	716

Selecting the right type of membership function - All aforementioned experiments were performed by selecting as the type of the membership function that the fuzzy sets follow, the trapezoidal one. A question arises whether the other two types of membership function that Fuzzy Miner supports, provide better performances upon the classification task. In order to answer this question the following two tables are provided, where in each column there is the mean value of the performance index (table 3) and the classification success (table 4) respectively. These averages are upon all the possible fuzzy partition sizes and they have been calculated for two values of alpha, where Fuzzy Miner presents relatively stable behaviour.

Table 3 Average PI vs membership functions

	Triangular	Trapezoidal	Gaussian
a =1	0.00556	0.0051	0.0040
a =5	0.00446	0.0042	0.0038

Table 4 Average CS vs membership functions

	Triangular	Trapezoidal	Gaussian
a =1	76.17	79.28	82.33
a =5	78.36	80.46	83.04

From the above tables, we draw the conclusion that the lowest performance index and the higher classification success occur when using the Gaussian membership function. The second best fitting is accomplished with the trapezoidal function. There is a logical explanation for the differences in the performances of these functions. First of all, the trapezoidal membership function is better than the triangular because trapezoidal function gives the maximum degree of compatibility (which is one) in more attribute values than the triangular function, which gives this maximum membership value just in those whose their value corresponds to the centroid of the triangular shape. As such, trapezoidal membership function gives higher degrees of compatibility in average, so the approximation of the desired output is becoming an easier task. Finally bell-shaped function is performing better than trapezoidal because it demonstrates a smoother transition between its various parts. Furthermore there is the possibility when using a trapezoidal membership function that some attributes are assigned the maximum degree of compatibility when they should be assigned lower degrees. This problem can be solved either by widening the big base or by narrowing the small base of the trapezoidal shape.

Missing rules - There is the possibility that Fuzzy Miner will not be able to predict an output for all input data pairs. This may occur if there is no rule in the rule base that corresponds to that input data pair. In the case of the simulations mentioned above, this problem occurred only for some specific parameter values, and particularly for large fuzzy partition sizes. The number of unpredicted outputs was very low (rarely more than 2). Nevertheless, this is also a criterion that must be taken into account when trying to optimize a fuzzy rule-based system.

5. Conclusions & Future work

This paper studies the pattern classification problem as this is presented in the context of data mining. More specifically, a fast heuristic fuzzy approach for classification of numerical data is described, followed by the design and the implementation of its corresponding tool (Fuzzy Miner). The approach does not need a defuzzification process; it can be utilized as a function approximator, while by slight changes can be used as a predictor rather as a classifier. The framework is highly flexible in that its components are configurable to meet various classification objectives. Linguistic representation of the produced fuzzy rules makes the classifier interpretable by native users, whereas the introduction of the adaptive procedure enables expanding and improving the rulebase while examining unseen, testing patterns. Fuzzy Miner was evaluated using the Athens Stock Exchange (ASE) data set. The strategy followed by Fuzzy Miner was proved successful and the results of the created classifier were shown.

Additional future work is planed in various aspects of Fuzzy Miner. To start with, pruning strategies could be used to improve the interpretability of the classifier. These pruning strategies can be either automatic or some control could be given to the user over the pruning process. Secondly, we have already started designing an algorithm for training the initially created fuzzy sets, by changing the length of the base or the height of a membership function, so representing the reality with greater precision. Additionally, in adaptive procedure, we can correct or discard some of the new rules, according to our pruning strategies. As such, it won’t be necessary to execute the pruning module for the whole rulebase from scratch, every time adaptive approach is used to improve the classifier. Furthermore, a potential expert user should be given the capability to initialize externally the rulebase, or to change existing rules produced by the algorithm and which do not agree with his domain knowledge. We can further help the expert by providing some statistics on the training data, before processing them. Another idea is to attach to the system an algorithm to automatically determine the number of fuzzy sets for each variable and a clear criterion of how “good” are the produced fuzzy sets. New GUI that supports graphical and textual displays (e.g. of the fuzzy sets) would be beneficial for interpreting the results of Fuzzy Miner. Finally, we plan to integrate Fuzzy Miner with a neural network and to propagate the outcome to a genetic algorithm that would extract the optimum solution upon a specific classification task.

Acknowledgements: This research was partially supported by the Greek Government, Ministry of National Education and Religious Affairs, under the EPEAEK II – Pythagoras Programme (2004-06).

References

[1] M. W. Craven and J. W. Shavlik, “Using neural networks in data mining”, Future Generation Computer Systems, 13:211-229, 1997.

[2] J. Han and M. Kamber, “Data Mining: Concepts and Techniques”, Morgan Kaufmann Publishers, 2001.

[3] H. Ichihashi and T. Watanabe, “Learning control system by a simplified fuzzy reasoning model”, Proc. IPMU’90 (1990) 417 – 419.

[4] B. Kosko, “Fuzzy systems as universal approximators”, Proc. FUZZ-IEEE ’92 (1992) 1153 – 1162.

[5] M. Mitchel, “An Introduction to Genetic Algorithms”, Cambridge, MA: MIT Press, 1996.

[6] V.S. Manoranjan, A. de Sam Lazaro, D. Edwards, and Aathalye, “A Systematic approach to obtaining fuzzy sets for control systems”, IEEE Transactions on Systems, Man and Cybernetics, Vol. 25, No 1, Jan. 1995.

[7] H. Nomura, I. Hayashi and N. Wakami, “A learning method of fuzzy inference rules by descent method”, Proc. FUZZ-IEEE ’92 (1992) 203 – 210.

[8] K. Nozzaki, H. Ishibuchi, H. Tanaka, “A simple but powerful heuristic method for generating fuzzy rules from numerical data”, Fuzzy Sets and Systems 86 (1997) 251 – 270.

[9] H. Nomura, I. Hayashi and N. Wakami, “A self tuning method of fuzzy reasoning by genetic algorithm”, Int. Fuzzy Systems and Intelligent Control Conf. (1992) 236 – 245.

[10] N. Pelekis, “*Fuzzy Miner: A Fuzzy System for Solving Pattern Classification Problems”*, M.Sc. Thesis, UMIST, 1999.

[11] R J. Schalkoff, “Pattern recognition: statistical, structural and neural approaches”, John Wiley and Sons 1992.

[12] M.Sugeno and G.T. Kang, “Structure identification of fuzzy model”, Fuzzy Sets and Systems 28 (1998) 15 – 33.

[13] M.Sugeno and T. Yasukawa, “A fuzzy-logic-based approach to qualitative modeling”, IEEE Trans. Fuzzy Systems 1 (1993) 7 – 31.

[14] H. Takagi and I. Hayashi, “NN-driven fuzzy reasoning”, Approximate reasoning 5 (1991) 191 – 212.

[15] T. Takagi and M.Sugeno, “Fuzzy identification of systems and its applications to modeling and control”, IEEE Trans. Systems, Man Cybernet. 15 (1985) 116 – 132.

[16] L.X. Wang and J.M. Mendel, “Generating fuzzy rules by learning from examples”, IEEE Trans. Systems, Man Cybernet. 22 (1992) 1414 – 1427.

[17] Zimmermann, H.-J., Hans-Jürgen, “Fuzzy set theory – and its applications”, H.-J. Zimmermann – 3rd ed. – Boston, Mass.; London: Kluwer Academic, 1996.