FUZZY MINER: EXTRACTING FUZZY RULES FROM NUMERICAL PATTERNS*

Nikos Pelekis[1,2,†], Babis Theodoulidis[2], Ioannis Kopanakis[2], Yannis Theodoridis[1]

[1]Information Systems Lab.
Dept. of Informatics, Univ. of Piraeus
Piraeus, Greece
URL: http://www.unipi.gr
E-mail: {npelekis | ytheod}@unipi.gr

[2]Center of Research in Information Management (CRIM)
Dept. of Computation, UMIST
Manchester, UK
URL: http://www.crim.org.uk
E-mail: babis@co.umist.ac.uk, kopanak@csd.uoc.gr

Abstract

We study the problem of classification as this is presented in the context of data mining. Among the various approaches that are investigated, 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. Such an approximation method coming from the domain of fuzzy control is appropriately adjusted into pattern classification and an 'adaptive' procedure is proposed for deriving highly accurate linguistic if-then rules. Extensive simulation tests are performed to demonstrate the performance of Fuzzy Miner, while a comparison with a neuro-fuzzy classifier of the area is taking place in order to contradict the methodologies and the corresponding outcomes. Finally, new research directions in the context of Fuzzy Miner are identified and ideas for its improvement are formulated.

Keywords: Pattern Classification, Data Mining, Fuzzy logic, Fuzzy Rules, Numerical Patterns.

FUZZY MINER: EXTRACTING FUZZY RULES FROM NUMERICAL PATTERNS

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 (Chen et al., 1996), is the search of relationships and global patterns that exist 'hidden' among vast amounts of data. There are various problems that someone has to deal with when extracting knowledge from data, including characterization, comparison, association, classification, prediction and clustering (Han & Kamber, 2001). This paper elaborates with the problem of *classification*. Broadly speaking, pattern classification (or recognition) is the science that concerns with 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 the 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 models, which are easier to be handled and more familiar to the human way of thinking.

After providing a brief comparative overview of pattern classification approaches (Section 2) and a short specification of the pattern classification domain in fuzzy systems (Section 3), the paper follows the above paradigm and describes an effective fuzzy system for the classification of numerical data (Section 4). The initial idea comes from the fact that fuzzy systems are universal approximators (Wang, 1992), (Kosko, 1992) of any real continuous function. Such an approximation method (Nozzaki et al., 1997) coming from the domain of fuzzy control systems is appropriately adjusted, extended and implemented 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 automatic deriving highly accurate linguistic if-then rules. The description of the methodology is combined with the illustration of the design issues of the tool *Fuzzy Miner*. The current work is evaluated (Section 5) by extensive simulation tests and by providing a comparison framework with another tool of the domain that employs a neuro-fuzzy approach, NEFCLASS (Nauck & Kruse, 1995). Finally, the paper concludes (Section 6) identifying promising directions for future work pointed to by this research effort.

COMPARATIVE OVERVIEW OF PATTERN CLASSIFICATION APPROACHES

Already, when the field was still in its very infancy, it was realized that statistics and probability theory (Berger, 1985) had much to offer to pattern classification (Schalkoff, 1992). 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 & fuzzy systems for pattern classification

Fuzzy logic (Zimmermann, 1996) 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"). Fuzzy Pattern Classification is one way to describe systems and the behaviour of systems. Computers always need exact data to process. With Fuzzy Pattern Classification we do not need such exact information. A system can be described by using adjectives like 'high', 'mid', 'low'.

Most applications of fuzzy systems can be found in the area of control engineering (fuzzy control). Fuzzy control applications are based on if-then rules. The antecedent of a rule consists of fuzzy descriptions of measured input values, and the consequent defines a - possibly fuzzy - output value for the given input. Basically, a fuzzy rule-based system provides an effective way to capture the approximate and inexact nature of the real world. In particular, fuzzy rule-based systems appear useful when the processes are too complex for analysis by conventional quantitative techniques or when the available information from the processes is qualitative, inexact or uncertain. Fuzzy rule-based systems have the theory of Fuzzy Logic as theoretical base. As stated in (Zimmermann, 1996), "fuzzy set theory provides a strict mathematical framework in which vague conceptual phenomena can be precisely and rigorously studied".

Fuzzy sets - A classical (crisp) set is normally defined as a collection of elements or objects $x \in X$, which can be finite or countable. Each element can either belong to or not belong to a set A,

$A \subseteq X$. Such a classical set can be described in different ways; either one can enumerate the elements that belong to the set, or one can define the member elements by using the characteristic function *l/A*, in which *1/A(x) = 1* indicates membership of *x* to *A* and *1/A(x) = 0* non-membership. Pattern Classification using fuzzy logic (Cios et al., 1998), (Manoranjan et al., 1995) 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.

Definition: If *X* is a collection of objects denoted generically by *x*, then a fuzzy set *A* in *X* is a set of ordered pairs:

$$A = \{(x, \mu_A(x)) \mid x \in X\} \tag{3.1}$$

μ_A is the membership function that maps *X* to the membership space *M* and $\mu_A(x)$ is the grade of membership (also degree of compatibility or degree of truth) of *x* in *A*. A widely used function is the so-called triangular membership function

$$\mu_{m,d}(x) = \begin{cases} 1 - \left| \frac{m - x}{d} \right| & \text{if } m - d \le x \le m + d \\ 0 & \text{if } x < m - d \text{ or } x > m + d \end{cases} \tag{3.2}$$

with $d > 0$ and $m \in \Re$. This function assumes the maximum membership degree of 1 at the value *m.* It decreases linearly to time left and right of *m* to membership degree 0. These fuzzy sets are suitable for modelling linguistic terms like *approximately zero.* Of course, the triangular function may be replaced by other functions, e.g. a trapezoidal or a Gaussian function.

Fuzzy rules - Fuzzy rules are a collection of linguistic statements that describe how a fuzzy inference system should make a decision regarding classifying an input. They combine two or more input fuzzy sets and associate with them an output. Fuzzy rules are always written in the following form:

> **IF** v_1 is A_1 **and** v_2 is A_2 **and** … v_n is A_n **THEN** $(v_1, v_2, \ldots, v_n)$ belongs to class *w,*
> where $A_1, A_2, \ldots, A_n$ are input fuzzy sets and *w* is output fuzzy set.

For example, one could make up a rule that says:

> **IF** temperature is high **and** humidity is high **THEN** room is hot.

There would have to be membership functions that define what we mean by high temperature ($input_1$), high humidity ($input_2$) and a hot room (output). This process of taking an input such as temperature and processing it through a membership function to determine what we mean by

'high' temperature is called *fuzzification*. The purpose of fuzzification is to map the inputs to values from 0 to 1 using a set of input membership functions.

The main advantage of the above connection is its close relation to human thinking. This is also the reason that the knowledge of an expert can easily be incorporated into a fuzzy pattern classification system. But in lack of an expert or in case of a complex system, there is also the possibility to use real information/data from the system to build the fuzzy rules. On the other hand the disadvantages are the necessity to provide the fuzzy rules, the fact that a fuzzy system cannot learn from data, and that there is no formal method to tune the membership functions.

The approaches introduced so far share some common features and common goals. Thus, boundaries between them are not very clear. Each has pitfalls and advantages and the availability and 'shape' of the features often determines the approach chosen to tackle a pattern classification problem. As far as fuzzy, statistical and structural approaches are concerned, all 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. If we were to classify the outcome of a coin flip, probability makes much more sense.

Neural networks and neuro-fuzzy systems

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 (Masters, 1993) - 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 (Bigus, 1996), (Craven & Shavlik, 1997).

Among the previously-mentioned solutions, fuzzy logic and neural networks can be an answer to the vast majority of classification problems. Both fuzzy systems and neural networks attempt to determine the transfer function between a feature space and a given class. Both 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, while a neural network, can only learn from scratch, and as a result it 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. Drawbacks still exist though as in example, the impossibility to extract rules from neurons for interpretation. As such, the training of the computer to optimize the classifier is usually much faster with a fuzzy classifier than a neural network classifier. Consequently, combining fuzzy logic and neural networks (neuro-fuzzy systems) we can avoid the drawbacks of each method. A neuro-fuzzy classifier of the area, called NEFCLASS, which is also used in the evaluation of our fuzzy system, has been introduced in (Nauck & Kruse, 1995).

Other approaches

The necessarily brief overview of the field would be incomplete without mentioning the existence of some alternative approaches, which are neither statistical nor syntactical. For example, the geometrical method (Prabhu, 2003) focuses on finding data representations or organizations, which are perceptually meaningful while the state-space method (Oja, 1983) is concerned with finding ways of searching effectively the hierarchical structures prevalent in many pattern recognition tasks. Furthemore, case-based reasoning methods (Leake, 1996), (Aamodt & Plazas, 1994), rough sets techniques (Pawlak, 1991), (Lenarcik & Piasta, 1997), (Swiniarski, 1998) and clustering methods (Liu et al., 2000) encompass diverse techniques for discovering regularities (or structures, or patterns) in complex data sets. They may serve to suggest either hypothetical models for the data-generating mechanism, or the existence of previously unknown pattern classes.

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 (Nomura et al., 1992), genetic-algorithm-based methods (Mitchel, 1996), least-squares methods (Sugeno & Kang, 1998), a fuzzy c-means method (Sugeno & Yasukawa, 1993) and a neuro-fuzzy method (Takagi & Hayashi, 1991). In (Wang & Mendel, 1992), an efficient rule generation method with no time-consuming iterative procedure is proposed.

FUZZY MINER: A FUZZY SYSTEM FOR SOLVING PATTERN CLASSIFICATION PROBLEMS

In this section, we present a powerful fuzzy system for solving pattern classification problems and we provide the reader with a description of the components of the Fuzzy Miner, their internal processes and their interrelationships. The reader interested in a more detailed description of the design and implementation issues of Fuzzy Miner is referred to (Pelekis, 1999). That work has mainly focused on the study and understanding of a method proposed in (Nozzaki et al., 1997). Fuzzy if-then rules with non-fuzzy singletons (i.e., real numbers) in the consequent parts are generated by the heuristic method proposed in (Nozzaki et al, 1997). In this paper, we innovatively adjust, extend and implement this function approximation method in order to produce an effective, working data mining tool in the field of pattern classification. A novel embedded '*adaptive*' process is also introduced, developed and incorporated into the previous mechanism for automatic deriving highly accurate linguistic if-then rules. The main advantage of these fuzzy if-then rules is the simplicity of the fuzzy reasoning procedure because no defuzzification step is required. The heuristic method determines the consequent real number of each fuzzy if-then rule 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.

Design and architecture of the fuzzy rule-based system

Fuzzy rule-based systems are also known as fuzzy inference systems, fuzzy models, fuzzy associative memories 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 the architecture depicted in Figure 1.

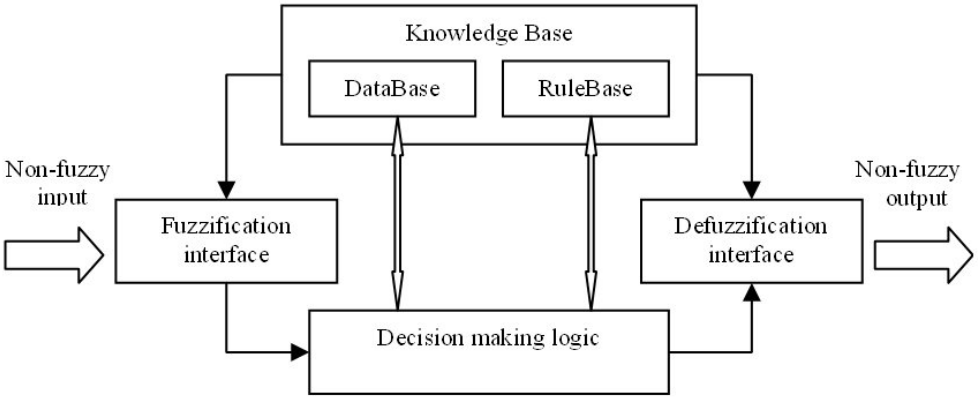

Figure 1 Architecture of Fuzzy Miner

In (Nozzaki et al., 1997), the authors consider a single-output fuzzy method in the *n*-dimensional input space $[0, 1]^n$, and we keep for the moment these assumptions just for simplicity reasons. The actual algorithm implemented in Fuzzy Miner 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 reversely map 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:

$$\{(x_p;y_p) \mid p = 1, 2, \ldots, m\} \tag{4.1}$$

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

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. On the other end, the defuzzification interface performs a mapping from the fuzzy output of a fuzzy rule-based system to a crisp output. However, the fuzzy rule-based system employed in this paper, does not require a defuzzification interface.

In the following subsections, we present in details the two core modules of the architecture, namely the knowledge base and the decision making logic.

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. In order to develop the appropriate infrastructure, Fuzzy Miner defines three corresponding parametric components, namely *Database*, *Fuzzy Partition* and *Membership Function*. *Database* 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* as the realization of a real database, which enables us to store, retrieve, update and generally manipulate data. The *Database* component 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 ith 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:

$$\left(A_{1j_1}, A_{2j_2}, \ldots, A_{nj_n}\right),\ j_1 = 1,2,\ldots,K_1,\ \ldots\ j_n = 1,2,\ldots,K_n \tag{4.2}$$

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 2(a). Figure 2(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.

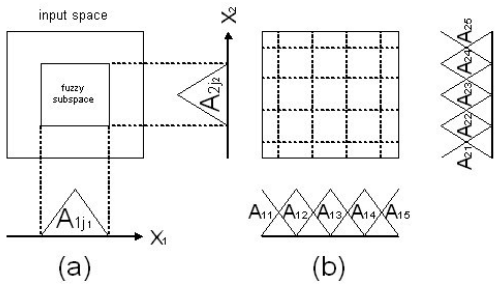

Figure 2 (a) Fuzzy subspace and (b) Fuzzy partition for $K_1 = 5$ and $K_2 = 5$

The *Membership Function* component 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. In order to be able to use more than one membership functions, we adopt 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. Figure 3 depicts the three types of membership functions that Fuzzy Miner supports.

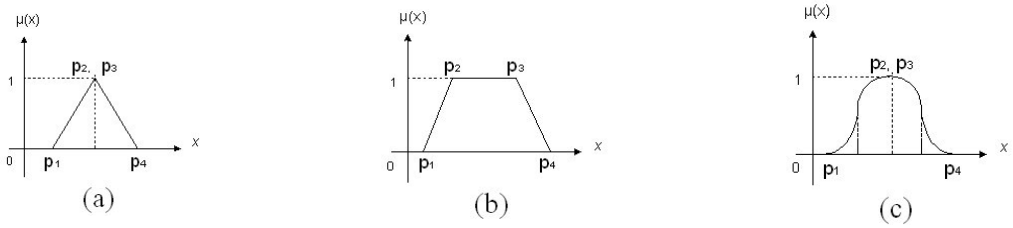

Figure 3 (a) Triangular, (b) Trapezoidal, (c) Bell-shaped

The *Fuzzy Partition* component 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 *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 *Fuzzy Partition* is a 2-D array of *Membership Functions* (Figure 4). 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.

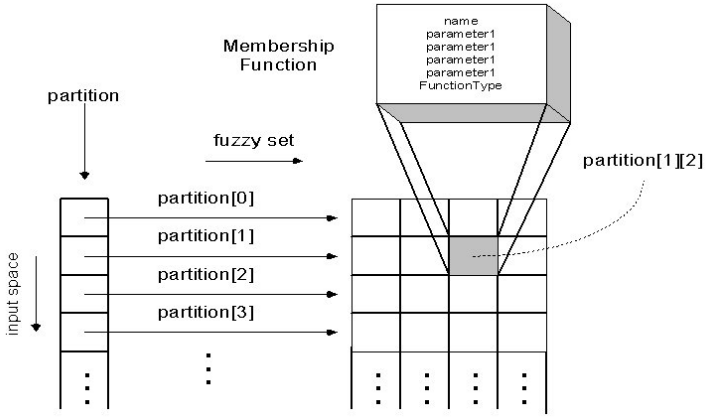

Figure 4 Fuzzy partition structure

The main functionality *Fuzzy Partition* offers Fuzzy Miner is the actual fuzzy partitioning taking place at the time of its initialization. More precisely, in order to create *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. Note that the training data set is considered, not the testing data set. This approach guarantees a minimum number of unpredicted outputs. 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. 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 $\Re$. In Figure 5, we present the partitioning in the case of a triangular membership function.

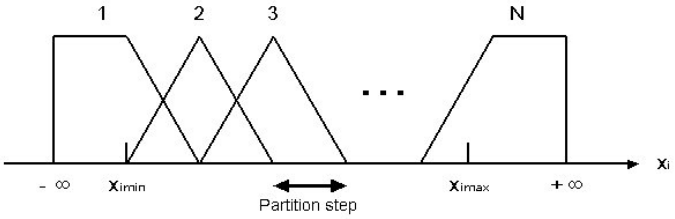

Figure 5 Fuzzy partitioning for triangular membership function

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 (4.3)

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 (Ichihashi & Watanabe, 1990), (Nomura et al., 1992). 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 (4.3), let us define the weight of the p_{th} input-output pair $(x_p;y_p)$ as

$$W_{j_1 \ldots j_n}(x_p) = \left\{\mu_{j_1 \ldots j_n}(x_p)\right\}^a, \tag{4.4}$$

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

$$b_{j_1 \ldots j_n} = \sum_{p=1}^{m} W_{j_1 \ldots j_n}(x_p) \cdot y_p \Big/ \sum_{p=1}^{m} W_{j_1 \ldots j_n}(x_p) \tag{4.5}$$

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 fuzzy rules from training data and furthermore is responsible for the decision making part of the algorithm (see section 4.1.2). An additional task 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* 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 4.5). 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 6.

Adaptive procedure - Before illustrating how the decision-making method has been developed, we further introduce an adaptive procedure with which we are capable of refining an existing rule base during its application upon a specific classification scenario. 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 idea 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 updated if a similar rule (same conditional part) exists in the rule base. By this action the consequent part of the rule is refined, by applying the generation method once more for this specific conditional part. Thus, the "adaptive" procedure enhances Fuzzy Miner with incremental characteristics. All available information is exploited, improving decision making on testing data.

```
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 6 Rule Generation Method

Linguistic representation **–** In real-world applications, it may be desirable that linguistic rules are generated from numerical data. In (Sugeno & Yasukawa, 1993) an approach for deriving linguistic rules from fuzzy if-then rules with fuzzy sets in the consequent parts is proposed. Here, a similar approach is adopted for translating fuzzy if-then rules with consequent real numbers into rules. "Then" part of such rules is a linguistic label and corresponds to the classification of the respective data pairs. 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 (4.3) 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) $B_1,B_2,...,B_N$, which are associated with the membership functions $\mu_{B_1},...,\mu_{B_N}$, 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 (4.3) are transformed to the following fuzzy if-then rules:

$$\text{Rule } R^*_{j_i\ldots j_n}: \text{If } x_1 \text{ is } A_{1j_1} \text{ and } \ldots \text{ and } x_n \text{ is } A_{nj_n} \text{ then } y \text{ is } B^*_{j_1\ldots j_n}, \text{ with } CF^*_{j_1\ldots j_n}, \quad j_1=1,2,\ldots,K_1;\ \ldots;\ j_n=1,2,\ldots,K_n, \tag{4.6}$$

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

$$\mu_{B^*_{j_1\ldots j_n}}\left(b_{j_1\ldots j_n}\right) = \max\left\{\mu_{B_i}\left(b_{j_1\ldots j_n}\right) \mid i = 1,2,\ldots,N\right\} \tag{4.7}$$

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

$$CF^*_{j_1\ldots j_n} = \mu_{B^*_{j_1\ldots j_n}}\left(b_{j_1\ldots j_n}\right) \tag{4.8}$$

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 fuzzy reasoning method of equation 4.9 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

$$y\left(x_p\right) = \sum_{j_1=1}^{K_1}\cdots\sum_{j_n=1}^{K_n}\mu_{j_1\ldots j_n}\left(x_p\right)\cdot b_{j_1\ldots j_n} \Big/ \sum_{j_1=1}^{K_1}\cdots\sum_{j_n=1}^{K_n}\mu_{j_1\ldots j_n}\left(x_p\right) \tag{4.9}$$

where $\mu_{j_1\ldots j_n}\left(x_p\right)$ 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 (4.6), which is given by

$$\mu_{j_1\ldots j_n}\left(x_p\right) = \mu_{1j_1}\left(x_{p1}\right)\times\ldots\times\mu_{nj_n}\left(x_{p_n}\right). \tag{4.10}$$

From (4.9), we can see that the inferred output $y(x_p)$ is the weighted average of the consequent real numbers $b_{j_1\ldots j_n}$'s of the $K_1K_2\ldots K_n$ fuzzy if-then rules.

Given a testing data set, this method 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. In order to improve results we utilize the adaptive procedure, which is an embedded process and not an 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) (eq. 5.2) and the number of unpredicted results are returned as the output of the whole process. This procedure is illustrated in Figure 7.

```
Decision Making ()
{
    construct & initialize an output database DB;
    construct & initialize one numerator & denominator per output;
    unpredictedResults = 0;
    create temporary rule;

    for(i = endOfLearning + 1; i <= endOfTesting; i++)
    {
            set IF part of temporary rule;
            set THEN part of temporary rule;
            calculate weight of temporary rule;
            re-initialize numerators & denominators per output;
            ruleFound = FALSE;

            for(k = 0; k <= NumOfRules; k++)
            {
                    estimate degree of compatibility of temporary rule with rulebase[k];

                    if (degree > 0)
                            for all outputs
                                    update numerators & denominators of temprule (using degree);

                    if (!ruleFound & temprule has same IF part as currentRule)
                    {
                            ruleFound = TRUE;
                            for all outputs
                                  update numerators & denominators of currentRule (using weight);
                    }
            }

            for all outputs
            {
                    write to output DB        //performing reverse mapping
                            1. original output
                            2. initial classification
                            3. inferred output
                            4. resulted classification
            }

            if (denominators != 0)
                    calculate performance index;
            else
                    unpredicted++;

            if (!ruleFound)
                    rulebase[++numOfRules] = temporary rule;
    }

    for all rules
            perform reverse mapping of inferred output to classes &
            set linguistic label for current rule;                    // classification result
            set probability of correctness for current rule;    // degree of certainty

    if (writeToDB)
    {
            write inferred data to database;
            if (denormalize)
                    if (normalize)     //normalization was performed on load data
                            denormalize output data;
    }
}
```

Figure 7 Algorithm for the Decision Making Engine

EVALUATION OF THE FUZZY MINER

In this section we focus on investigating 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 output. As such, in order to measure the performance of the methods implemented in Fuzzy Miner, we have run several experiments that result in interesting conclusions. These experiments have also been compared to the results derived from another classifier, which uses a neuro-fuzzy

approach (Nauck & Kruse, 1995). For the experiments we used the data set from the **A**thens **S**tock **E**xchange (ASE, 2004) 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. The calculation of this real number as the result of a function is based upon the following information that the ASE database stores:

- *Max price*: the maximum point in the fluctuation of the price of a stock during a day
- *Min price*: the respective minimum point in the above-mentioned fluctuation
- *Exchanged items*: the number of stocks that were sold/bought during a day
- *Close price*: the ending point in the daily fluctuation of the price of the stock

Our choice for such a function is a formula that takes into account three factors and estimates a real number in the interval [0, 3]. More specifically, each of these factors calculates a normalized number from zero to one, indicating of how much a specific stock fluctuated during a day. Based on these sub-formulas, the overall indication of the fluctuation of the stock is derived. Therefore, the function that estimates the consequent (output) real number, which Fuzzy Miner will try to infer/approximate, is described in equation 5.1:

$$f : \Re^4 \to [0,3] \text{ and } f(x_1, x_2, x_3, x_4) = factor1 + factor2 + factor3 \tag{5.1}$$

$$f(x_1, x_2, x_3, x_4) = \frac{(Maxprice - Minprice) - MINdiff}{MAXdiff - MINdiff} + \frac{(Closeprice - Minprice) - MINdiff}{MAXdiff - MINdiff} + \frac{ExchangedItems - MINitems}{MAXitems - MINitems}$$

where x_1, x_2, x_3, x_4 are the four input parameters of the data set.

The first factor indicates the fluctuation of the difference between the maximum price and the minimum price of a stock. *MAXdiff* and *MINdiff* are the maximum and minimum (respectively) differences in the data set between *Max price* and *Min price* attributes. The second factor models the fluctuation of the difference between the closing price and the minimum price of a stock. This is an indication of how easily a stock maintains its price away from the minimum price. *MAXdiff* and *MINdiff* are the respective maximum and minimum differences between *Close* and *Min price* columns. Finally, the third factor tries to strengthen the two previous factors by adding to them the normalized number of exchanged items. By this, we model the fact that if the fluctuation of a stock is high, then this is more important when the number of exchanged items is also high, than when

the number of stocks that where sold or bought is low. *MAXitems* and *MINitems* are the maximum and minimum values of *Exchanged items* attribute.

Experimenting with Fuzzy Miner

In order to assess the forecasting ability of Fuzzy Miner we limited our experiments to the banking sector from where we sampled 3.000 input-output data pairs collected from the daily transactions of eight banking constitutions, during a calendrical year (1997). A set of 1.500 tuples were used for learning and the remaining 1.500 tuples for testing. Note that, since the fuzzy rule-based system can employ the 'adaptive' procedure, test data may be learning data as well, although they do not participate in the creation of the initial fuzzy rule base.

Fitting & generalization ability for training and testing data

For the evaluation of the algorithm the summation of square errors, between the desired output y_p and the inferred output $y(x_p)$ for each input-output pair $(x_p; y_p)$ is calculated. This performance index (PI) for Fuzzy Miner is given by the equation 5.2:

$$PI = \sum_{p=1}^{m} \{y(x_p) - y_p\}^2 / 2 \tag{5.2}$$

The two most important factors of the fuzzy rule-based system are the value of α and the size of the fuzzy partitions. In order to understand the influence of these parameters on the performance index, the algorithm has been invoked with different values of α, varying from 0.1 to 50, and a fuzzy partition size varying from 2 to 25. The results of the simulations are presented in Table 1, which contains only a subset of the experimentations. Note that for each different value of α we first present PI without using the adaptive approach, and subsequently PI using the adaptive procedure.

Fuzzy sets \ α	*2*	*3*	*4*	*5*	*6*	*7*
0.1	0.013042	0.005884	0.005064	0.003990	0.003728	0.003467
	0.012900	0.005830	0.005028	0.003912	0.003518	0.003380
0.5	0.011439	0.005519	0.004895	0.003954	0.003716	0.003464
	0.011340	0.005453	0.004866	0.003884	0.003477	0.003378
1	0.009960	0.005212	0.004719	0.003915	0.003713	0.003467
	0.009905	0.005117	0.004702	0.003864	0.003435	0.003392
5	0.006252	0.004578	0.004198	0.003719	0.003249	0.003558
	0.006147	0.004452	0.004268	0.003817	0.003157	0.003688
10	0.005421	0.004470	0.004292	0.003664	0.003294	0.003132
	0.005626	0.004452	0.004212	0.003767	0.004056	0.003988
50	0.005767	0.004848	0.004583	0.003972	0.003937	0.004712
	0.005598	0.005090	0.004789	0.004050	0.004989	0.005109

Table 1 Performance index against α & number of fuzzy sets

An obvious conclusion that could be inferred from Table 1 is that larger sizes of fuzzy partition lead to a better fitting (smaller PI) to the given input-output data pairs. Using the table containing the complete results of the previously mentioned simulations, the PI for both the original method and the method using the adaptive approach have been plotted against the number of fuzzy sets per fuzzy partition. Figure 8 indicates the strength of the adaptive approach. However, when $\alpha > 1$, PI might become worse due to the phenomenon of overfitting. Further investigation designates that when the number of fuzzy sets is high PI decreases very slowly, whereas for a small number of fuzzy sets PI is much more sensitive to the variation of the fuzzy partition size. Finally, PI asymptotically converges to a specific limit as the fuzzy partition size increases.

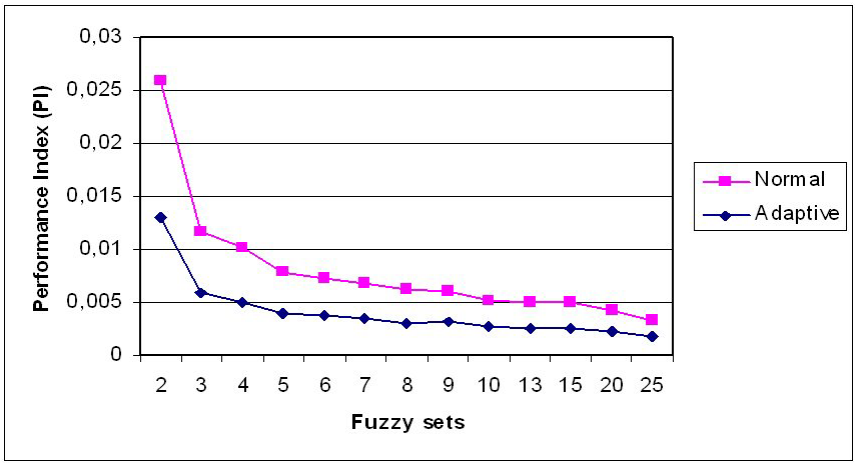

Figure 8 PI against size of fuzzy partitioning

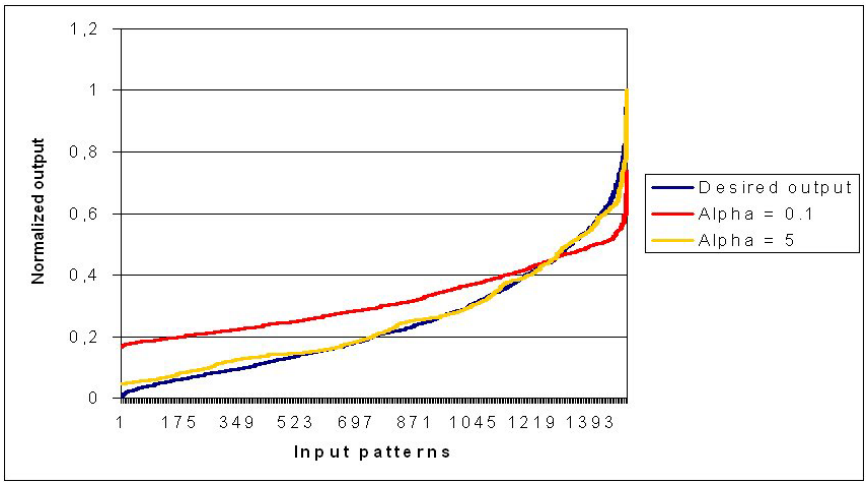

Figure 9 Fluctuation against α

An additional observation is that for each specific fuzzy set PI decreases or increases depending on the value of α. More specifically, when α is less than five, PI is improving, but when α exceeds that limit, PI starts decreasing. The best fitting is presented when $\alpha = 5$. As a conclusion we could argue that PI can be improved by choosing the appropriate value of α. Figure 9 illustrates the

desired output and two inferred outputs, for two different values of α. When α = 5, it is self-evident that the approximation of the formula is much better than when α is 0.1.

Classification success

In order to investigate the classification success of Fuzzy Miner with respect to different sizes of the fuzzy partitioning and the values of α, the following table is provided.

Fuzzy sets \ α	*2*	*3*	*4*	*5*	*6*	*7*
0.1	92.8632	88.7925	76.3175	74.5831	70.4470	67.1114
	94.8632	90.4603	76.3175	74.9833	69.5797	66.4443
0.5	93.9300	88.7258	77.1181	75.1835	70.9139	67.3783
	94.9967	90.5270	77.1514	75.3169	69.9133	66.3109
1	95.8301	88.3923	78.0520	75.4503	71.1141	67.5784
	95.9310	90.5935	79.6518	76.3175	70.3135	66.5777
5	96.3302	88.1921	79.9867	77.1181	73.7158	68.4456
	96.3302	90.8606	79.3863	77.3849	70.1134	66.7760
10	94.9967	88.4590	81.0540	77.3849	73.4490	67.5117
	94.9967	90.7272	79.8532	77.9853	69.4463	64.4430
50	93.2234	87.3249	79.6531	75.4503	71.8479	61.7078
	94.5297	89.7265	79.4530	76.0507	66.7111	61.3075

Table 2 Classification success against α & number of fuzzy sets

A conclusion that accords with the conclusion inferred previously is that when the number of fuzzy sets is fixed, then for values of α lower than five the percentage of classification success increases as α approximates five. When it exceeds five, the trend is either to stabilize or to decrease. This conclusion does not stand strongly as in the case of PI. This is reasonable due to the vagueness that is introduced by the fuzzy sets. There is the possibility that PI of the classifier can be improved without a corresponding improvement of the classification success. Table 3 presents the trend of the classifier for the case of two fuzzy sets (classes).

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

Table 3 Classification accuracy against α

The previous reason also explains why Table 2 includes cases where the percentage of success is the same when using the adaptive approach and not. Except for those few situations where the two percentages are identical, the general trend that is followed is that for a 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 improve performances. (By overfitting, we mean the

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 easier but also the stronger inference that someone could make from Table 2 is that increasing the number of classes results in decreasing the percentage success of the classifier. This conclusion is illustrated in Figure 10, which shows that low fuzzy partition sizes have as a consequence a rapid reduction of the classification success. On the contrary, for a high number of fuzzy sets we observe a stabilization in the rate of reduction of the classifier.

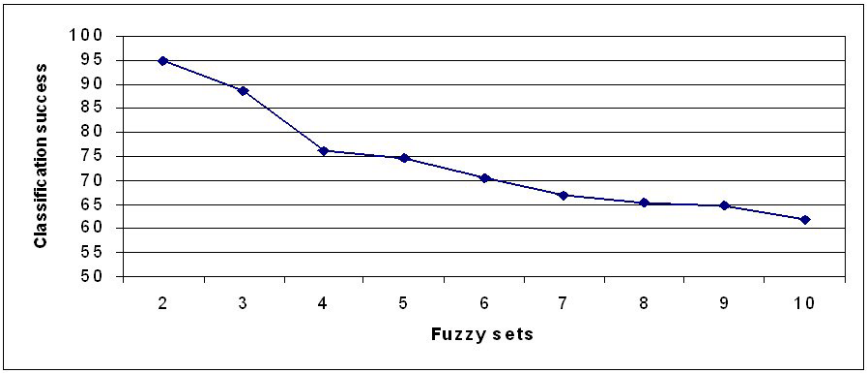

Figure 10 Classification success against size of fuzzy partitioning

Optimizing the size of the Rule Base

Another interesting observation that someone could notice by running the Fuzzy Miner with the previous parameters, is the variation of PI with respect to the size of the rule base. More specifically, Figure 11 indicates that when 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 be also determined. The relation between the fuzzy partition size and the inferred fuzzy rules is shown in Figure 12.

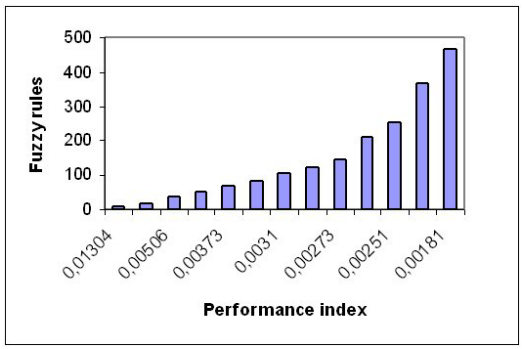

Figure 11 Size of rule base against PI

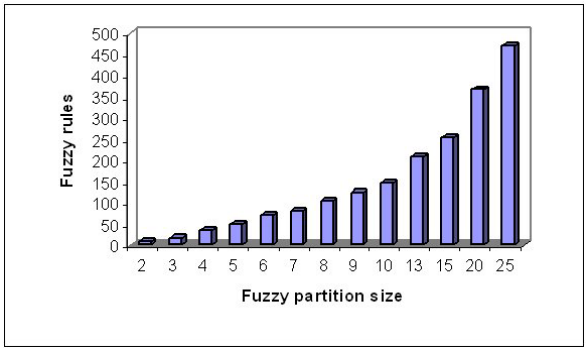

Figure 12 Rule base against fuzzy partition size

The fact that the number of the produced rules is exponential interdependence of the number of fuzzy sets used to partition the input space, could also be inferred from Table 4, where the number

of rules for different sizes of fuzzy partition is presented. Table 4 includes an extra row containing the number of rules when the adaptive approach is applied. As expected, the size of the rule base gets larger as new rules are added during the decision making stage.

# Fuzzy sets	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

Table 4 Produced rules for different numbers of fuzzy sets

Selecting the right type of membership function

All the above experiments were performed by selecting the trapezoidal membership function. Legitimately, the enquiry if the other two types of membership function that Fuzzy Miner supports provide better performances upon the classification task arises. In order to answer this question, the following two tables are provided, where each column presents the mean value of PI (Table 5) and the classification success (Table 6), respectively. The presented averages are upon all possible fuzzy partition sizes and have been calculated for two values of α, where Fuzzy Miner presents relatively stable behaviour.

	Triangular	*Trapezoidal*	*Gaussian*
$\alpha = 1$	0.005567	0.005118	0.004069
$\alpha = 5$	0.004462	0.004259	0.003801

Table 5 Average performance index for different types of membership function

	Triangular	*Trapezoidal*	*Gaussian*
$\alpha = 1$	*76.1704*	*79.2862*	*82.3326*
$\alpha = 5$	78.3643	80.4647	83.0439

Table 6 Average classification success for different types of membership function

From the above tables, we draw the conclusion that the lowest PI and the higher classification success are derived 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 the former gives the maximum degree of compatibility (which is one) in more attribute values than the latter, which gives this maximum membership value just in those 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. Unfavourably there is the possibility when using the 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. Finally, bell-shaped function has an improved behaviour as it exhibits a smoother transition between its various parts.

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 simulations performed, this problem occurred only for some specific parameter values, and particularly for large fuzzy partition sizes. The number of unpredicted outputs was very small (rarely more than 2). Nevertheless, this is also an additional criterion that must be taken into account when trying to optimize a fuzzy rule-based system.

Comparing Fuzzy Miner with NEFCLASS

Continuing our evaluation, we have chosen NEFCLASS (Nauck & Kruse, 1995) as the method for contradicting our fuzzy system, as it has many common features and it shares the same goals as Fuzzy Miner. We have repeated experiments on NEFCLASS similar to those that were used to explore the validity of Fuzzy Miner. In these experiments we are interested in the particular characteristics of NEFCLASS and in this section we present a summary of the results of the simulations performed. Table 7 reviews the experiments that took place.

When NEFCLASS uses…	*Classification success*
The default parameterized method (triangular function)	79.7346
Trapezoidal membership function	80.0426
Gaussian membership function	80.9283
Cross validation procedure	92.0244
The *"best"* rule learning method	82.9870
The *"best per class"* rule learning method	81.0765
The training of fuzzy sets approach	85.5647
Pruning strategies after default method	85.7442
Pruning strategies after cross validation	93.8734

Table 7 Classification accuracy of the NEFCLASS model

Comparing the classification success of NEFCLASS for its various parameters with the corresponding percentages from the third column of Table 2 (number of fuzzy sets equal to three), we can have a clear picture of the cases that Fuzzy Miner performs better. In general, Fuzzy Miner gives higher classification performances than NEFCLASS. The only case where NEFCLASS classifies patterns with a higher rate is when it uses the cross validation procedure either autonomously or in conjunction with one of the pruning strategies. In the second situation the success of the classifier is even higher. The explanation for this is that cross validation procedure uses the whole pattern set both as training and as testing data set, and performs several validation

steps in order to create a classifier with the best fitting upon this specific data set. Another conclusion is that both NEFCLASS and Fuzzy Miner have slightly better classification results when using the Gaussian membership function.

The main difference in the philosophy of generating fuzzy rules between NEFCLASS and Fuzzy Miner is not in the algorithmic part of the two approaches. Both algorithms utilize congener methods to produce the antecedent part of a rule but they diversify in the calculation of the consequent part of the rule, by employing different heuristic methods. The crucial difference that disjoins the two approaches is that NEFCLASS by using the *"best"* rule and *"best per class"* rule learning method diminishes the size of rulebase significantly. In addition to this, someone could decrease the number of the produced rules even more by employing one of the supported pruning strategies. On the other hand, our approach searches the whole pattern space and produces rules so every training input pattern is satisfied by some rule. This high number of rules is further increased by introducing the adaptive approach, which creates new rules when test data are processed. This major difference between NEFCLASS and Fuzzy Miner is emanation of the different philosophy and intention of the two applications. NEFCLASS concentrates on the conciseness and the readability of the fuzzy rules, whilst Fuzzy Miner tries to cover all possible input patterns. That is actually the main explanation why Fuzzy Miner outperforms NEFCLASS in the classification of unseen data.

There is a significant increment in the performance of NEFCLASS when using the method of training the fuzzy sets that were initially constructed by the rule generation method. By training the fuzzy sets, we mean that the base or the height of the initial membership functions is adjusted, so the fuzzy sets can partition the pattern space in a better way. This improved way of fuzzy partitioning means that the degree of membership of a specific input value to a fuzzy set becomes larger or smaller according to some user-defined criterion. Fuzzy Miner also supports a very simple way of tuning the membership functions, which is actually the α parameter. This form of tuning the membership functions after a certain point causes overfitting to data and the classification accuracy of the classifier is worsened. Another problem with improving the shape of the membership functions is that the change that is performed is uniform, so it cannot take into account the uneven distribution that is a common case in real world data. The optimum solution is try to capture this unevenness of the data set and transmit it to the shape of the membership functions (e.g. scalene triangles).

Both NEFCLASS and Fuzzy Miner employ linguistic representation methods to derive fuzzy if-then rules. The main difference between these rulebases is their size. NEFCLASS produces less and smaller rules but they are more readable and concise. On the other hand, Fuzzy Miner has the disadvantage that could extract many rules for huge data sets, but it has the advantage that, by

taking into account the whole pattern set, it can show important exceptions or trends in the data set. From the point of view of the developer such rules may seem useless, but from the point of view of an expert these rules may identify exceptional behaviours or problematic cases. As such, NEFCLASS is more likely to lose useful extracted knowledge by excluding whole rules or parts from a rule.

An additional issue that should be mentioned is that the generation of a rulebase in Fuzzy Miner is almost instantaneous (several seconds for large data sets). On the contrary, in order to improve the initial created classifier, NEFCLASS has to train the fuzzy sets, prune the rule base and repeat the decision making stage. All these operations result in a more readable rulebase but the whole process is time-consuming. Furthermore, Fuzzy Miner has the advantage that it can classify more than one output concurrently, based on the same set of input attributes. In order to do this in NEFCLASS, one has to create different input files for each output and to rebuild new classifiers from scratch. Last but not least, in Fuzzy Miner several configurations can be tried and the best one can be selected on the basis of the lowest test error. If the number of unpredicted outputs is not zero, a coarser fuzzy partition should be tried. On the other hand, this is not necessary in NEFCLASS as all these are automatically performed by the cross validation procedure.

CONCLUSIONS & FUTURE WORK

This paper studies the pattern classification problem as this is presented in the context of data mining. More specifically, an efficient fuzzy approach for classification of numerical data is described, followed by the design and the implementation of its corresponding tool, called 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, 2004) data set. The strategy adopted by Fuzzy Miner was shown to be successful and the results of the created classifier were presented.

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 could 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 refine or discard some

of the new rules, based on the indications of our pruning strategies. As such, it won't be necessary to execute the pruning module for the whole rulebase from scratch, every time the adaptive approach is used to improve the classifier. Furthermore, a knowledge expert should be provided with the capability to initialize the rulebase externally, or to change existing rules which do not agree with his domain knowledge. We can further help the expert in the preprocessing stage by providing some statistics on the training data. 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. A new user interface that supports graphical and textual displays (e.g. of the fuzzy sets) would be beneficial for interpreting the results of Fuzzy Miner (Kopanakis & Theodoulidis, 2003). Finally, a long term goal is 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.

---

* A short version appears in the informal Proceedings of the 1st Int'l Workshop on Pattern Representation and Management (PaRMa'04), Heraklion-Crete, Greece, March 2004.
† Contact author's address: 80 Karaoli-Dimitriou St., GR-18534 Piraeus, Greece. Tel: +30-2104142449, Fax: +30-2104142264.

REFERENCES

Aamodt, A., & Plazas, E., (1994). Case-Based reasoning: Foundational issues, methodological variations, and system approaches. AI Comm., 7:39-52.

ASE (2004). The Athens Stock Exchange closing prices. Available at: http://www.ase.gr/content/en/MarketData/Stocks/Prices/default.asp. Current as of 26/3/2004.

Berger, J. O. (1985). Statistical Decision Theory and Bayesian Analysis. Springer-Verlag. 2nd edition.

Bigus, J. P. (1996). Data Mining with Neural Networks – Solving Business Problems. McGraw-Hill.

Chen, M.S., Han, J., & Yu, P.S. (1996). Data Mining: An overview from a database perspective. IEEE Transactions on Knowledge and Data Engineering, 8(6): 866-883.

Cios, K., Pedrycz, W. & Swiniarski, R. (1998). Data Mining Methods for Knowledge Discovery. Kluwer Academic Publishers.

Craven, M. W. & Shavlik, J. W. (1997). Using neural networks in data mining. Future Generation Computer Systems, 13:211-229.

Han, J. & Kamber, M. (2001). Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers.

Ichihashi, H. & Watanabe, T. (1990). Learning control system by a simplified fuzzy reasoning model. In Proceedings of IPMU'90, 417 – 419.

Kopanakis, I. & Theodoulidis, B. (2003). Visual Data Mining Modelling Techniques for The Visualization of Mining Outcomes. Journal of Visual Languages and Computing, Special Issue on Visual Data Mining, 14(6): 543-589.

Kosko, B. (1992). Fuzzy systems as universal approximators. In Proceedings of FUZZ-IEEE '92, 1153 – 1162.

Leake, D. B. (1996). CBR in context: The present and future. In D. B. Leake (ed.), Case-Based Reasoning: Experience, Lessons and Future Directions, AAAI Press, p. 3-30.

Lenarcik, A. & Piasta, Z. (1997). Probabilistic rough classifiers with mixture of discrete and continuous variables. In T. Y. Lin and N. Cercone (eds.), Rough Sets and Data Mining: Analysis for Imprecise Data, Kluwer Academic Publishers, p. 373-383.

Liu, B., Xia, Y., & Yu, P. (2000). Clustering through decision tree construction. In Proceedings of *ACM CIKM*.

Manoranjan, V.S., Lazaro, A. de Sam, Edwards, D., & Aathalye, (1995). A Systematic approach to obtaining fuzzy sets for control systems. IEEE Transactions on Systems, Man and Cybernetics, 25(1).

Masters, T. (1993). Practical Neural Network Recipes in C++. Academic Press.

Mitchel, M. (1996). An Introduction to Genetic Algorithms. MIT Press.

Nauck, D. & Kruse, R. (1995). NEFCLASS – A neuro-fuzzy approach for the classification of data. ACM Press.

Nomura, H., Hayashi, I. & Wakami, N. (1992). A learning method of fuzzy inference rules by descent method. In Proceedings of FUZZ-IEEE '92*,* 203 – 210.

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

Oja, E. (1983). Subspace methods for Pattern Recognition. John Wiley.

Pawlak, Z. (1991). Rough Sets, Theoretical Aspects of Reasoning about Data. Kluwer Academic Publishers.

Pelekis, N. (1999). Fuzzy Miner: A Fuzzy System for Solving Pattern Classification Problems. M.Sc. Thesis, UMIST. Also available at http://users.forthnet.gr/ath/pele/ HOME_PAGE_NIKOS_PELEKIS/Download/. Current as of 15/6/2004.

Prabhu, N. (2003). Gauge groups and data classification. Applied Mathematics and Computation, 138(2-3): 267-289.

Schalkoff, R. J. (1992). Pattern recognition: statistical, structural and neural approaches. John Wiley.

Sugeno, M. & Yasukawa, T. (1993). A fuzzy-logic-based approach to qualitative modeling. IEEE Trans. Fuzzy Systems, 1: 7–31.

Sugeno, M.& Kang, G.T. (1998). Structure identification of fuzzy model. Fuzzy Sets and Systems, 28: 15–33.

Swiniarski, R. (1998). Rough sets and principal component analysis and their applications in future extraction and selection, data model building and classification. In S. Pal and A. Skowron (eds.), Fuzzy Sets, Rough Sets and Decision Making Processes, Springer-Verlag.

Takagi, H. & Hayashi, I. (1991). NN-driven fuzzy reasoning. Approximate Reasoning, 5: 191–212.

Wang, L.X. & Mendel, J.M. (1992). Generating fuzzy rules by learning from examples. IEEE Trans. Systems, Man Cybernet, 22: 1414–1427.

Wang, L.X. (1992). Fuzzy systems as universal approximators. In Proceedings of FUZZ-IEEE'92, 1163–1170.

Zimmermann, H.-J. (1996). Fuzzy set theory and its applications, Kluwer Academic, 3rd edition..